Inverse Illumination Design with Genetic Programming

Kelly Moylan

Department of Computer Science

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of

Master of Science

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

For everyone I know, and everyone I don't know.

Abstract

Interior illumination is a complex problem involving numerous interacting factors. This research applies genetic programming towards problems in illumination design. The Radiance system is used for performing accurate illumination simulations. Radiance accounts for a number of important environmental factors, which we exploit during fitness evaluation. Illumination requirements include local illumination intensity from natural and artificial sources, colour, and uniformity. Evolved solutions incorporate design elements such as artificial lights, room materials, windows, and glass properties. A number of case studies are examined, including many-objective problems involving up to 7 illumination requirements, the design of a decorative wall of lights, and the creation of a stained-glass window for a large public space. Our results show the technical and creative possibilities of applying genetic programming to illumination design.

Acknowledgments

This work would not have been possible if not for the constant support and help from my family, friends, students, and mentors. I would like to specifically acknowledge the following people/organizations for their contributions:

- Brian Ross, for his guidance, support, and patience. There is no better mentor.
- My parents, Susannah and Barry, for supporting and encouraging my endeavor into even more schooling.
- Jennifer, for everything.
- Matt, and all my other friends for keeping me going and keeping me me.
- Cale Fairchild, for without his technical wizardry, most of this would not have been possible.
- Brock University and the Computer Science Department, for the use of their resources, funding, and facility.

It's been a blast.

K.J.M

"If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas.

– George Bernard Shaw

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The design of interior spaces is a task which has always required a sharp creative mind, as well as an understanding of how lighting can affect and change a room. For the most part, these two aspects complement each other, and are required for all parts of a design to work together. Within any space, the use of both natural and artificial light sources is required, as almost all rooms must be able to be occupied during both day and night. This gives the task to the designer of using both sources to their maximum efficiency. Most well trained interior designers will intuitively understand the illumination needs of a room, building an understanding of illumination design from years of experience[33]. This can lead to the creation of great atria, with sides completely made of glass that fill the room with brilliant sunlight, while saving those inside from harsh brightness and glare. These same rooms can then be illuminated during the night, with artificial lighting designed to create completely different moods and settings. The aesthetic nature of this task means that describing exactly how a designer creates these layouts is difficult to do precisely, as our understanding of the technical side of aesthetics is still growing. The use of computers in this problem is worth serious consideration.

When looking at the use of lighting, many aspects which a designer takes into consideration would be difficult to describe and measure with a computer, such as the moods which are invoked from different shading or the purpose of the room which can be defined. While these areas may not be able to be accomplished by a computer, some design decisions can be. The measuring of illumination is one aspect which will always hold importance to designers. If the functional goal of that space cannot be accomplished due to poor lighting choices, the space itself becomes unusable. Designers take extra care when looking at the placement of lights in rooms to make sure that the desired effects will be feasible.

The creation of an automated program which could help with illumination design would be of great use to designers. While a program would not be a replacement for the skill and creativity of a human designer, it could be used as an aid in giving new inspiration and concept ideas to the designer from which they can expand. The creativity that a computer based design program can bring is not constrained by conventional ideas of what should be done or the learned habits of those who have experience. The system would also be able to take a purely algorithmic approach of optimality based on given criteria. New ideas can help inspire whole new ways of looking at lighting design, and that is a tool which these people would find very valuable.

The problem of inverse illumination is directly tied with the area of evolutionary design that looks at energy efficiency. The sun is an energy source which has great utility not only for interior design, but practical properties of buildings. A direct relation between lighting and energy efficiency can be drawn along these lines. A focus on energy efficiency can add new complexities to the problem, and allows researchers to share information from both.

A 3-dimensional structure designed for occupation must consider many factors. Artificial and natural light need to work together as to not make either redundant, and also suit the requirements set forth by the designer. These requirements can be based on many different human factors, needing different amounts of lighting for different purposes[29]. Work environments will require a lower amount of lighting to reduce glare off computer screens. Wide open social spaces will require higher amounts of light to create pleasant environments to spend time in. Construction material and paint, and even decorative objects, will have an effect on the reflection and spreading of light.

Geographic location will have a significant effect on the production of a structure as well. A building cannot simply be lifted and transplanted to another location and be expected to have the same desired feel. Northern and southern hemisphere differences will dictate window placement. Some structures are designed with specific times of day or year in mind. Stonehenge is said to be an ancient calendar based on the sun. Wanting certain lighting conditions at noon must factor the movement of the sun across the sky throughout the year.

All these factors are complex and detail oriented on their own. Combining them together to create a specific desired outcome creates a challenging problem to solve.

1.1 Goals and Motivations

Our goal is the development of an automated system which can be used for designing illumination solutions for predefined structural layouts. The use of genetic programming (GP) techniques is considered[23], as this approach is new to the inverse illumination research field.

This thesis has three main goals to be accomplished. The first is to demonstrate the effectiveness of GP techniques on this problem. The next goal is to consider numerous illumination factors, to increase the complexity of the problem. The final goal is to examine the application of procedural texture generation in illumination design. The inverse illumination problem has been studied before with different computational intelligence techniques. GP can offer a great degree of flexibility over the different illumination design.

Most work previously done in this area has focused on a single element of the problem, whether it be the positions of light objects, or window elements, or any other factor. While each of these factors are important in their own right, consideration of their interrelationships is a new level of complexity for the problem. The addition of more elements will show the increased difficulty of this problem.

Another goal of this research is to use the knowledge from evolutionary texture generation towards solving problems in decorative illumination design. By using lighting analysis for an evaluation technique, a non-direct measurement is being used, where traditionally direct measurements on the texture were used. This could be a new and interesting technique of texture creation with real world properties and applications.

1.2 Thesis Structure

The thesis is structured as follows. Chapter 2 gives detailed background information on the different subjects of this work. This includes information on evolutionary design, genetic programming, inverse illumination, and the RADIANCE simulation system. Chapter 3 provides a literature review of related works in the subjects of evolutionary design, energy efficiency, and illumination design. In Chapter 4, the specification of the system used for this work is detailed in full, including fitness evaluation, system architecture, and language.

Chapter 5 presents basic experiments done as a proof of concept for our work. Work done specifically focused on windows and natural lighting is outlined in Chapter

6. Chapter 7 explores decorative applications using texture generation and light. Chapter 8 discusses the results of this thesis. Finally, Chapter 9 presents conclusions and future possible work.

Chapter 2

Background

This chapter will introduce the required background information for understanding the research presented. This includes genetic programming as well as illumination and the Radiance system.

2.1 Genetic Programming

Genetic programming (GP)[23] is a branch of evolutionary computation , which is an area of artificial intelligence which uses a simulation of biologically inspired evolution for solving complex problems. GP applies Darwinian evolution by natural selection to evolve a population of possible solutions to a problem, based on the fitness of each individual. The more fit an individual is, the better suited it is for survival, and the more likely it is to pass on its genetic material to the next generation. In the

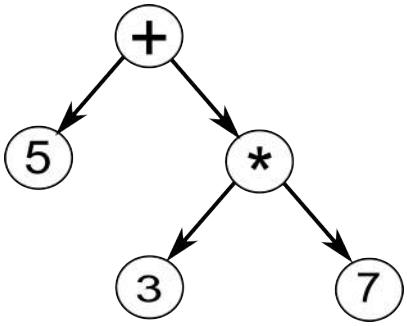

Figure 2.1: Example GP tree

most common form of GP, solutions are represented in tree form. Figure 2.1 shows an example of the representation. Nodes of the tree are called functions, and the

leaves are known as terminals. The terminals are usually the data which the tree will manipulate to arrive at its final answer. This manipulation is done by the functions, which could range from basic arithmetic operations to complex calculations.

The basic algorithm for a GP is given in Algorithm 1. What follows in this subsection is an explanation of each piece of the algorithm.

```
Initialize random population;
Evaluate initial population;
while Termination condition not met do
    Add most fit individual to next generation;
    while Next population is not full do
        Select two random individuals from population based on fitness;
        Perform crossover and mutation on selection and add to next
        generation;
    end
    Evaluate new population and assign fitness values;
end
Return results and data from runs;
```

Algorithm 1: Genetic Program algorithm pseudo code

2.1.1 Initialization

During the initialization phase, the first population is created. Each candidate is created at random and added to the initial population. The generation of these candidates is controlled by the minimum and maximum tree depth parameter to stop the generation or programs from growing too large, as well as being large enough.

2.1.2 Candidate Evaluation

At the end of each generation, each member of the population is evaluated to determine its fitness score. This score is determined by the evaluation of the tree through a fitness function. This fitness function is specifically designed for the problem at hand.

2.1.3 Fitness-Proportional Selection

Tournament selection is used here to choose which candidates are used in breeding to create the next generation. In tournament selection, k candidates are selected at

random. The fitness of these candidates are then compared, and the best is selected for the first parent. The process is then run again to select the second parent. By keeping the tournament size low, usually between 3 and 5, more diversity in the breeding selection, as it is less likely to select the same candidate multiple times.

2.1.4 Reproduction - Crossover and Mutation

The reproduction stage consists of two operators - crossover and mutation. Both of these operations are done to fulfill the exploration and exploitation of the search space required for the algorithm to be successful.

Crossover involves taking two chromosomes from the population and exchanging pieces between them. With tree based GP systems, the most common type of crossover is sub-tree crossover, as seen in Figure 2.2. Using sub-tree crossover, a node is selected in each of the candidates at random. The two sub-trees are exchanged, and the newly created chromosomes are added to the next generation's population.

Mutation is the other breeding operation, and it is applied to the newly created

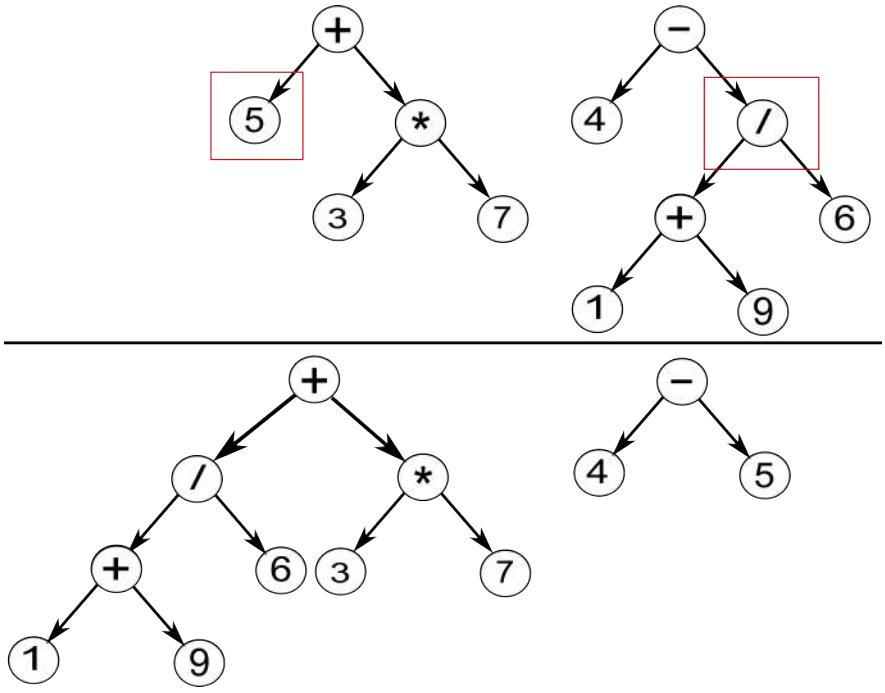

Figure 2.2: Example crossover and resulting children

chromosomes. One common mutation technique is the sub-tree mutation, as seen in Figure 2.3. One node is randomly selected from the tree, and the entire branch is replaced with a new randomly created sub-tree. While this has the possibility of

making major changes to the chromosome, because of the branching nature of GP trees, the change is more likely to be small.

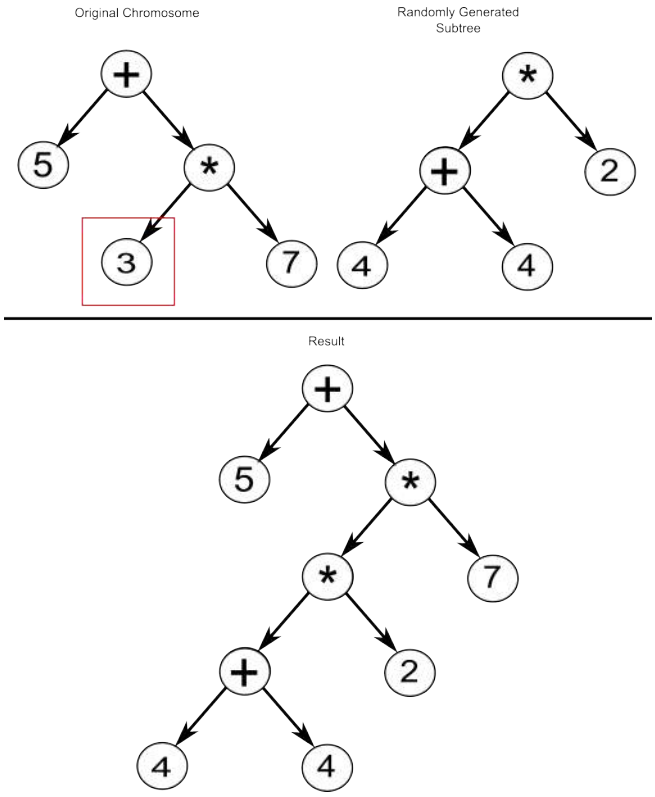

Figure 2.3: Example mutation and resulting child

2.1.5 Elitism

To keep the population fitness stable, the best n candidates of each generation can be carried over directly to the next. Because of the random nature of GP, it is entirely possible to lose good candidates during the reproduction phase. Elitism guarantees that the next generation will have a candidate at least as good as the current generation. This ensures that the best fitness will not decrease.

2.2 Many-Objective Optimization

In complex problems, there is normally more than one goal which is trying to be satisfied. Multi-objective optimization techniques look to find the best ways to optimize multiple features simultaneously[17]. Each feature will have its own target, and the

value of these features are all taken into account in some way. When working with multiple objectives, it is never clear which objective has the most value, and therefore some form of equal measurement must be considered. For all many-objective problems considered in this thesis, normalized ranked sum is used.

2.2.1 Weighted-Sum

Weighted-Sum is a ranking system in which the various objectives are susceptible to the most amount of bias[8]. Each objective is measured in its raw form, and these values are then added together to give an overall fitness score towards that candidate. Because the importance of objectives is not distributed equally, a modifier is assigned to each individual value. This can be expressed as

$$fitness = \sum_{i=1}^{n} v_i \times m_i \tag{2.1}$$

where n is the number of objectives, v_i is the fitness value of objective i, and m_i is the modifier assigned to objective i. Although this is the simplest method for multi-objective evaluation, the need to determine weighing for each objective individually can introduce bias into the evaluation.

2.2.2 Pareto Ranking

Pareto ranking works on the notion that unless a solution is better in every aspect, it is impossible to say if one is better than another[17]. It seeks to remove any bias towards one objective over another, and does so by ranking candidates based on tiers. Each tier is better in every aspect then the tier below it, but each solution in the rank cannot be said to be better than the others. Each tier is made by finding the Pareto front, which is the group of the best solutions that are not dominated by any other solution. Dominating is defined by:

$$\forall i : A(i) \leq B(i) \wedge \exists i : A(i) < B(i) \tag{2.2}$$

This means in a minimization problem, A dominates B if all elements within A are less than or equal to all elements within B, and there exists at least one element within A which is less than its counterpart in B. Any non-dominated candidates are removed from the population and given a rank of one. The process is then started again, and non-dominated candidates are given a rank of two. This is repeated until

all candidates are assigned ranks. This creates a grouping of individuals who are all considered as good as the next in their tier. Pareto ranking has limitations, decreasing in effectiveness in problems with greater than 5 objectives.

2.2.3 Normalized Ranked Sum

Normalized Ranked Sum[3, 8] is used for problems with 4 or more objectives. This is because Pareto ranking begins to fail at high dimensional problems of this size.

Normalized Ranked Sum (or Average Rank) is an effective many-objective strategy. Each candidate in the population is evaluated. Each objective in the candidate is ranked based on its comparison to the values of the same objective in each other candidate. Each rank is normalized by the number of unique ranks which have been assigned for that objective. This ratio is then added with the other ratios for the solution, and the sum is used as the new overall rank of the candidate. The fitness is found through

$$fit = \sum_{i=1}^{k} \frac{r_i}{maxRank_i}$$

where r_i is the rank of the objective at position i in the rank vector $\vec{R} = (r_1, ..., r_k)$, and $maxRank_i$ is the maximum number of ranks for the objective i.

Table 2.1 gives an example of this process. Each objective in the candidates are ranked based on the numbers in the other candidates. In standard ranked sum, the candidates are ranked 4,3,1,2,2, based on the sum of their ranks. When the ranks are normalized, the tie between the two final candidates is removed.

Fitness	**Ranking**	**RS**	**NRS**
(24,0,45,12)	(4,1,5,2)	12	2.83
(0,22,44,32)	(1,3,4,3)	11	2.8
(13,0,21,6)	(3,1,3,1)	8	1.93
(3,11,13,32)	(2,2,2,3)	9	2.06
(0,22,0,48)	(1,3,1,4)	9	2.45

Table 2.1: Ranked sum example

2.3 Evolutionary Design

Evolutionary design is the study of the application of evolutionary computation to the design of forms, models, and structures[2]. The use of algorithms for the design of structures and forms has been used for many years[22], automating their algorithms on computers in the next steps. The goal of this automation of design is not to replace designers and architects, but the creation of aids as well as for bolstering human inspiration from evolved techniques. Unlike many areas of focus in evolutionary computation, much research in evolutionary design is not directly searching for an optimal solution. Many problem spaces which are searched could have multiple satisfying solutions.

There are two common approaches to evolutionary design: interactive evolution and autonomous evolution. Interactive evolution is often used for evaluating evolved designs[39]. A human judge is used to determine the quality of the solutions. This is the easiest to implement but also can introduce bias in the evolution through the human interaction as well as user-fatigue when evaluating lots of candidates. Autonomous evolution allows the design process to happen completely independently. This can allow the system to explore more possible options and express creativity, but requires a greater understanding of design factors that are not completely understood. Without encoding expert knowledge of the subject, the evolved solutions can lack some of the qualities which a human designer would deem necessary.

One area of design which can be considered a difficult problem is the qualitative properties of the design process. Most work in evolutionary design focuses on the quantitative evolution of design. Working with numbers is easier than trying to translate aesthetics and emotion. However, the area of computational aesthetics is a new area of research[4] attempting to translate these aspects.

2.4 Illumination Design

2.4.1 Inverse Illumination

The inverse illumination problem involves finding potential lighting solutions for a pre-defined environment[40]. This consists of deciding different aspects of lighting: number of lights, intensity, shape, position, etc. Not only are artificial sources under consideration, but the use of natural lighting can be integrated as a factor as well. The interaction of light between objects also matters[7]. A common characteristic of illumination problems is that while many aspects of the scene are unknown at

the beginning, the desired final illumination requirement of the scene are known in advance. This means the final desired appearance of the scene or environment can be used to work backwards to establish the missing parameters[31].

A major development in the measurement of illuminance for this area of research came from Nabil and Mardaljevic[29] with the development of the Useful Daylight Illuminance (UDI). While it maintains much of the simplicity of the conventional daylight factor approach, which is a measurement of the brightness of sunlight inside a building compared to outside, it builds a foundation on detailed studies of the visual comfort of varying light levels. The UDI looks at an annual measurement of the amount of sunlight which falls between an acceptable range of 100-2000 lux. This range was derived through the study of multiple published works on the comfort of differing light levels as well as the codes and standards for various nations around the world. This metric influenced later illuminance research.

There are two main areas of research in inverse rendering problems: inverse lighting problems, and inverse reflectometry problems[31]. This thesis is will be focused on the former. These problems include inverse emittance, where the emittance of light sources are unknown, and inverse light positioning, where optimization lies in the locations of light sources to achieve a desired illumination. Inverse reflectometry problems are ones in which the reflective property of a surface is unknown and must be defined for its effect on the environment.

2.4.2 Radiance

Radiance[25] is a program designed for the analysis of lighting, which has been photometrically and photo realistically validated[26]. This means that it has been shown to accurately simulate the properties of light, through comparison with the real world[19]. Developed with the support of the U.S. Department of Energy and the Swiss Federal Government, it is a widely used and proven system for the simulation and analysis of light and spaces. Its use in research has ranged from the study of single bedrooms to solar radiation absorption of urban cities[36, 38]. It is capable of rendering environments in full 3-D from user generated information. The software itself is a light weight UNIX based system, which allows the user a very high level of control over many environmental factors. Factors include the material of which objects are created and different levels of control over light sources, with position, colour, and intensity all user controlled. The system includes controls for time of day for a full day analysis of a space, as well as location in the world. All of these

controls over the environment make Radiance an extremely useful and flexible tool for studying illumination factors and design in a 3-D environment.

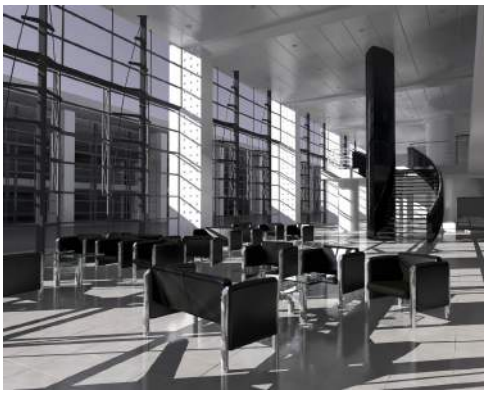

Figure 2.4: Example scenes created in Radiance[43]

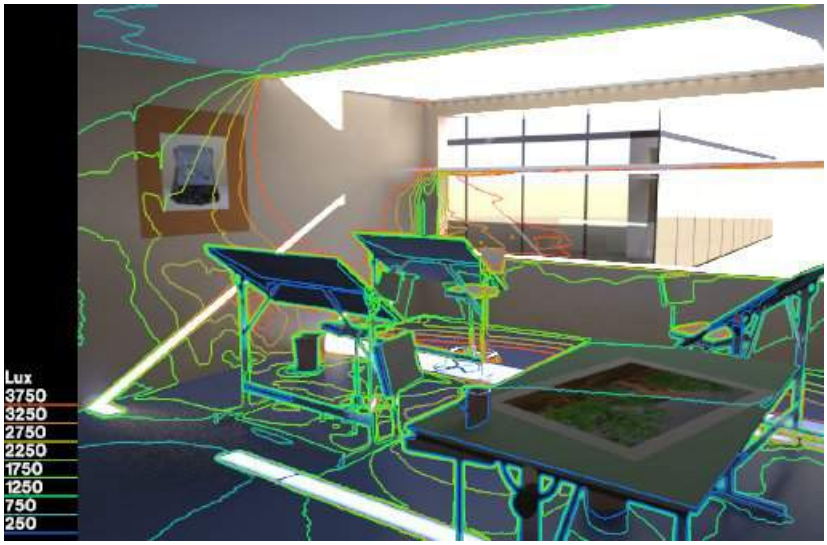

Figure 2.5: A false colour image in Radiance used to gather lux values[24]

The simulation engine of Radiance uses a hybrid approach of Monte Carlo radiosity and deterministic ray tracing. This creates rendered scenes which are extremely realistic. The image in Figure 2.4 is an example of a scene rendered with Radiance. The software pioneered of the concept of high dynamic range imaging (HDR), where light levels are open ended, rather than the typical bounded values of standard imaging. A form of tone mapping is used in this work[44]. Radiance includes a number of tools specifically designed for lighting analysis, including the false colour images to show the levels of luminosity throughout a scene. Figure 2.5 is a false colour image

showing the lux levels on the surfaces. These images are visual representation of the differences in illumination levels upon surfaces. Glare indices of surfaces and sources of glare are used to calculate the Guth visual comfort probability, which is a measure of the percentage of a population which with find a scene comfortable. Radiance is used as a backbone for many lighting simulation software, including DAYSIM, a daylighting analysis tool which models the annual sunlight exposure in and around buildings.

The sky model of Radiance is set to Commission Internationale de l'Éclairage (CIE) standards, recreating the both the overcast standard and clear sky model. Adopted in 1955 as a standard, this model is one of the most commonly used for illumination design [26]. Overcast sky conditions are used most often for the comparative evaluation of designs, predictions of daylight effects from exterior obstructions, and architectural analysis for compliances. The CIE Standard Overcast Sky has had its validity shown through comparisons of measured data to demonstrate its accuracy in representing dull sky conditions[21].

Chapter 3

Literature Review

3.1 Computational Intelligence and Illumination

Evolutionary research in the field of illumination design has explored a wide range of analysis and measurement techniques.

Tena[40] was one of the first to apply a genetic algorithm (GA) to the problem. He used an interactive system, where the desired solution was specified on each surface in the scene. To minimize the difference between the solution and the desired target, the genetic algorithm was integrated to find the best lighting configuration. Evaluated by the use of hierarchical radiosity, the user of the system was able to define all parameters of the scene. The selection of the best scenes guided the algorithm towards the target.

Fernandez *et al.*[13] took the approach of a heuristic search to recreate an initial lighting scene. The program was tasked with finding placements of skylights, as well as experiments in combination with artificial light sources, in a predefined space. Their approach used the VNS meta heuristic, with successive explorations of neighbourhoods. This allowed a wider search space as the system moved towards the target optima. Two different types of experiments were performed to test the capabilities of the heuristic. The first moved a set number of light sources in the ceiling, trying to position for a reflective radiosity target. The second experiment was a multi-objective problem of maximizing the light power of the scene, while minimizing the power of the artificial lights. This also included a minimization of light on certain target areas of the room.

Costa *et al.*[9] took a similar approach to Fernandez *et al.*, optimizing a scene with inputs of geometry, material property, and design goals. Optimization was performed using simulated annealing, chosen for its ability to process cost functions

with arbitrary degrees of conditions and constraints. Using Radiance to perform lighting calculations, optimization would continue based on the set parameters until full met.

Mourshed *et al.* looked at both artificial and natural lighting using Radiance for lighting analysis[36]. Using a hospital patient room as a test case, geometric parameters of a window and light shelf were optimized towards daylight factor as well as heating and cooling. Using modelling and analysis software, an exhaustive search of all possible solutions in the search space was performed. In a second study[35], the placement of an artificial light source in a senior living space was studied. Unlike the exhaustive search before, a GA was implemented for the optimization. A grid of measurement points were distributed over the horizontal and vertical surfaces of the room and used for illuminance measurements. The GA optimization process proved to be successful in design decisions.

Shea *et al.*[34] took a unique approach with the use of Ant Colony Optimization (ACO) to design panelled building envelopes. In their work, the ceiling of a structure was divided into a grid pattern, and the ACO was used to determine the panel material for each space on the grid. Using three unique types of glass and an opaque panel, they set to optimize the daylight factor as well as the amount of direct sun hours, while minimizing the amount of opaque panels used. This would cause a favour of solutions where the glass panels would not contribute to the sun hours measured. The work was further used to develop a prototype GUI the use of optimizing more building envelopes.

Work by Culter *et al.*[10] considered fenestration material in window design. The refractive properties of glass plays a major role in the use of the material, and new developments in glass creation are tested in an interactive setting.

3.2 Computational Intelligence and Energy Efficient Lighting

Caldas[6] investigated automation for energy efficiency, using a GA without interactive fitness. Her system, GENE_ARCH, is a generative design system for use by architects. GENE_ARCH was used to design building structures with relation to many different factors, including materials, construction costs, energy use, and thermal and lighting behaviour.

Castro *et al.*[7] used a number of different heuristic and evolutionary based ap-

proaches for finding energy-saving light positioning. These techniques solved illumination designs at optimally minimum emission power. The work serves as a comparison between standard evolutionary computation methods and new memetic approaches, which combined evolutionary algorithms with local optimizations. Using a hill climbing heuristic in combination with a GA and particle swarm optimization, they looked at improving proven methods which have had wide use in the field of illumination research. They found that while the memetic approach will yield better solutions, there is a major trade-off in computation time.

Oraei looked at the use of passive solar energy from a purely energy efficiency perspective[16]. Focusing on the use of passive solar energy, using the sun for heating and cooling properties has a direct relationship with the use of the sun in inverse illumination. Having a room be filled with sunlight is a desired trait, but the affect it has on the temperature system of a building can cause many more issues. The work used (GP) to develop energy efficient buildings with respect to heating and cooling costs, accounting for window design and placement, roof design, building materials, and size of building. Other factors considered were geography and location, as well as time of day.

Ghobad *et al.*[15] analysed the effects of the geometry of skylights on passive solar lighting. Their work tested the effect the standard skylight design with a splayed aperture, as well as integrating structural components for a space saving skylight design. Evaluating the daylight factor, as well as the UDI, their study showed a dramatic increase in performance between the standard design and the splayed aperture. They also looked at the energy efficiency effect this performance had on the rooms, again finding a major correlation between the use of passive solar energy and energy efficiency. This work showed the effectiveness evaluating the geometry of the building in conjunction with the standard illumination approaches.

Marin *et al.*[27] studied passive solar energy relating to architecture and structure of buildings rather than the interiors. Marin considered envelope design during the early stages of development of a structure. They stressed the creative abilities of architectural program and the assistance they may be able to provide to professionals.

The work of Turrin *et al.* [42] also examines the outer shell of buildings, addressing the design of large roof structures for semi-outdoor structures. They improved daylight and thermal comfort, and explored of the use of passive solar energy to reduce artificial light and control temperature. Both of these studies employed the use of GAs to encourage creativity in the exploration of solutions, as well as interaction with a designer to set parameters and requirements.

3.3 Evolutionary Textures and Graphics

Dawkins[11] demonstrated that Darwinian evolution had potential application in the field of graphics. Using 2-D graphic objects called biomorphs, he used interactive evolution and computation to allow a user to guide evolution of graphic structures to represent desired shapes.

Sims[37] used evolutionary techniques to create complex structures, textures, and motions in computer graphics. Using interactive selection, he was able to evolve 3-D plant structures by evolving procedural parameter sets, as well as 2-D and 3-D textures using symbolic expression evolution. He also evolved animations, using time as a variable in the genotype.

Graf and Banzhaf[18] used interactive GP for evolving 2-D bitmap images as well as 3-D voxel structure images. They used morphing and warping to breed together bitmap images for a user to select through image interpolation. Linear transformations were used on the 3-D voxel images to create variation images for evolution.

Todd and Latham[41] evolved computer sculptures made with constructive solid geometry techniques. They used a selective mutation technique with interactive evolution to create aesthetic sculptures.

Ross *et al.*[32] used an unsupervised GP system with multi-objective fitness to evolve procedural textures. Using a feature test based on empirical analysis of fine art, which had shown that many analyzed art works had a bell curve distribution of colour gradients. This model was found to create pleasing images to users, while measures without the model created boring or chaotic images.

Heijer and Eiben[12] compared four different aesthetic measures and their application with evolutionary art. They used unsupervised evolution using GP to create visual art, and hand picked some of the most fit and best images to verify their fitness measures. They showed that each aesthetic measure had its own "style" of image it created, and that some measures can be very inflexible in terms of creativity.

In relation to stained glass generation, Brooks[5] presented a method of restyling an image to resemble the appearance of a work of stained glass. Input images are segmented and transformed to match segments of real stained glass taken from a database. This technique was able to produce high quality results with little user interaction.

Ashlock *et al.*[1] took an evolutionary approach to the generation of stained glass styled images. Using Voronoi tiles, the centres are evolved to minimize the variance of luminance within each tile. A fractal model is run to create a texture based on the

colouring of the tiles. Lead edging is added to finalize the stained glass appearance, then the textures are applied to a final image.

Other aspects of design have been investigated for optimization in recent years, including colour harmony[28] , graphic design layouts[30], and arrangements of furniture[45].

Chapter 4

System Specifications

4.1 Overview

The goal of this system is to automate the creation of a room with GP, focused on the implementation of many objectives. The design of a language to describe these aspects is key. Fitness is taken through four different means of measurement. The GP system used in our research is the RobGP system[14], a C++ based GP system. This system was chosen for the ease of integration with the Radiance simulation system.

4.2 Architecture

The first step of execution is the loading of the RobGP parameters into the system. This includes the population size, generation count, tree depth parameters, and breeding parameters. Random GP trees are then generated to begin the evolution process. When the tree is evaluated, terminal node values are used to execute Radiance system commands. These commands will modify a Radiance input file. Radiance can be used with one or multiple input files. Here, any part of the environment which was not changeable by the GP system was separated into predefined model structures. Once the tree has finished execution, the created Radiance input files are compiled along with any other necessary Radiance files into an oct-tree (OCT) file. This OCT file is used to generate the described model environment and is used by the Radiance system for any analysis required. This OCT file would then be analyzed using Radiance's internal light analysis tool set, rtrace. The fitness evaluation uses this created file to gather the required values for the experiment being performed.

Figure 4.1 shows the interaction between they two systems used, and Algorithm 2

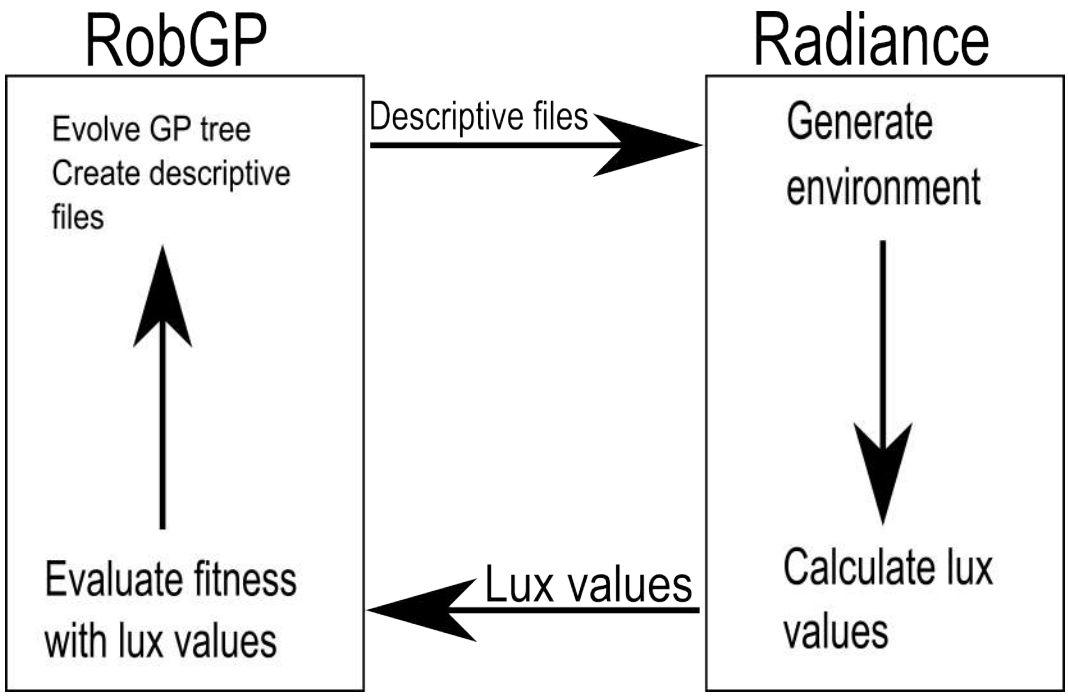

Figure 4.1: System execution

is a summary description of the execution. The population is ranked based on fitness values, and the termination condition is checked. If the termination condition (solution found or max generation reached) is not met, the breeding process is performed, a new population is created for the next generation, and the system restarts are the tree evaluation step. If the termination condition is met, evolution is stopped. Then the required statistics and best found solution are generated, along with the Radiance files which are used to describe this solution.

4.3 Genetic Programming Language

Table 4.1 shows the strongly typed language of the GP system. This typing defines all the aspects of the room to be defined, as well as allowing multiple methods of creating aspects of the room to be selected.

The function set of the system is described in Table 4.2. The functions used are detailed in the list below, with more specific explanation of some advanced functions given in their specific chapters:

- *Root(W,MM,LM,LW,SG)* is the root of the GP tree. Each component of the room is kept in a separate sub-tree, allowing for the addition or removal of

```
Load RobGP parameters;
Initialize random population;
Generate description files from population trees;
Compile OCT file from description files;
Generate environment and analyse lux values using Radiance rtrace command;
Use found lux values to evaluate population and assign fitness;
while Termination condition not met do
    while Next population is not full do
        Select two random individuals from population based on fitness;
        Perform crossover and mutation on selection and add to next
        generation;
    end
    Evaluate new population and assign fitness values;
end
Return results and data from runs;
```

Algorithm 2: Execution pseudo code

Symbol	**Representation**	**Description**
R	Root	Top level of tree structure
LM	Light Maker	Top level of light creation
W	Windows	Top level of creating windows on walls
MM	Material Maker	Top level of creating wall materials
C	Ceiling	Top surface of the room
NW	North Wall	Wall located at the north end of the building
SW	South Wall	Wall located at the north end of the building
EW	East Wall	Wall located at the north end of the building
WW	West Wall	Wall located at the north end of the building
NM	North Material	Material of north wall
SM	South Material	Material of south wall
EM	East Material	Material of east wall
WM	West Material	Material of west wall
CM	Floor Material	Material of ceiling
FM	Floor Material	Material of floor
M	Material	Material definition
LB	Light bulb	Artificial light source
LW	Light Wall	Creation of wall of lights texture
SG	Stained Glass	Creation of the stained glass textures
I	Integer	Integer value
F	Float	Float value
TF	Tree Function	Set of mathematical functions for texture trees

Table 4.1: GP Language Types

Type	Function Name	Description
R	`Root(W,MM,LM,LW,SG)`	Creates the scene. Parameters vary according to design task.
LM	`Top_Light(LB[n..m])`	Creates between n and m artificial lighting objects.
LB	`Basic_Light(F,F)`	Light source of fixed white intensity. Args define location.
W	`Windows(C,NW,SW,EW,NW)`	Creates walls and windows.
MM	`Materials(NM,SM,EM,WM,CM,FM)`	Creates materials.
C	`SkyLight(F,F,F,F)`	Skylight window on the ceiling. Args define coords for 2 opposite corners.

NW
SW
EW
 	
NW
	`North_Wall_Center(I,F,F,F,TF)`
	`South_Wall_Center(I,F,F,F,TF)`
	`East_Wall_Center(I,F,F,F,TF)`
	
	`West_Wall_Center(I,F,F,F,TF)`
		Patterns the walls of the room based on percentage measures for window size and
		location on wall sections.

NM
SM
EM
WM
CW
FW
	`North_Material(M)`
	`South_Material(M)`
	`East_Material(M)`
	`West_Material(M)`
	`Ceiling_Material(M)`
	`Floor_Material(M)`
		Creates the material used for each surface in the room and uniquely identifies them for creation.

M	`Material(I,F,F,F,F,F)`	Material defn: R, G, B, reflection, roughness.
LW	`Light_Wall(I,I,TF,TF,TF)`	Grid of lights on a wall. TF expressions compute RGB of each light.
SG	`Stained_Glass(I,I,TF,TF,TF)`	Grid of stained glass on wall.
TF	Add(TF[2..4])	Add op for colour expressions. Between 2 to 4 arguments.
TF	-, *, /, neg, sin, cos, log	Other math operators.
TF	X, Y	Grid coordinates.
TF	`ERCTF`	Ephemeral TF ($-1.0 \leq TF < 1.0$)
F	`ERCFloat`	Ephemeral float ($0.0 \leq F < 100.0$)
I	`ERCInt`	Ephemeral integer ($0 \leq I < 100$)

Table 4.2: Function Set

specified sub-systems.

- *Top_Light(LB[n..m])* will create up to m light objects, where n and m are a user defined minimum and maximum number of light objects which can exist in the scene.

- *Basic_Light(F,F)* will create a light object at position [F,F]. A check is completed to determine if the incoming object is too close, a distance of 1 metre, to another existing object or the walls, as well as not being placed in the same area of the skylight.

- *Windows(C,NW,SW,EW,NW)* is the root of the window creation sub-tree. This separates each wall to have individual control and windows, as well as the sky light.

- *Materials(...)* assigns materials for the 4 walls, ceiling, and floor. If not used, pre-defined materials are assigned.

- *SkyLight(F,F,F,F)* Skylight window on the ceiling. Arguments define x and y coordinates for 2 opposite corners.

- *Direction_Wall_Center(I,a,b,c,d)* creates windows for the given *direction* wall. "*I*" is converted to a value between 0 and 30, and is the number of wall panels or sections to create for windows. The a, b, and c arguments use the fraction portion of the float value. They scale the windows. All the windows on a wall's panels will have the same scale. The d argument determined whether a window is to be created on a panel. Its floating point expression is given the panel coordinates. If the value is positive, a window is defined on that panel. Otherwise no window is created.

- *Direction_Material(M)* is a function used to separate the different surfaces and allow for them to develop unique materials. The *direction* corresponds with one of the six available surfaces for use: north, south, east, and west walls, as well as floor and ceiling.

- *Material(I,F,F,F,F,F)* will create a material definition for its assigned surface. An integer is used to determine which of three material choices will be used: plastic, metal, or mirror. Five floating point values correspond to red, green, blue, reflection, and roughness. The plastic and metal material definition are

defined the same, where as the mirror only uses the first three floating point values for red green and blue.

- *Light_Wall(I,J,R,G,B)* generates a K-by-L grid of lights. I and J are integers converted to values between $3 \leq K \leq 10$ and $3 \leq L \leq 36$. Each light's grid coordinate is accessible to the colour channel expressions.
- *Stained_Glass(I,J,R,G,B)* creates a grid of square stained glass windows on a wall. It works much like *Light_Wall(I,J,R,G,B)*. One difference is that we project each glass element's coordinate to the range $-1.0 \leq x, y \leq 1.0$. This range can be altered as desired.

For the experiments, not all functions of the language are used at the same time, though they could all be used together. Functions are selected based on the needs of the experiment.

4.4 Fitness Evaluation

Four separate fitness measures are used in the experiments:

- Luminosity Intensity
- Luminosity Evenness
- Glare
- Colour

4.4.1 Luminosity Intensity

All measurements of light in Randiance are completed in three separate colour channels: red, green, and blue. The following equation is used

$$179 * (.265 * Red + .670 * Green + .065 * Blue)$$

to translate the the colour channel measurement into a standard white light based reading. This is the standard conversion equation of irradiance to illuminance used in Radiance. This is done to make sure the measurements are done with accuracy to real world measurements, as the three channels would be viewed as a single wave length. This value is used as an intensity measurement of the light. Many experiments

held certain requirements of light intensity for differing areas of the room, and this measurement was used.

4.4.2 Luminosity Evenness

Because the intensity of the light is measured from multiple points, it would be possible for the value to get dominated by bright spots while leaving an unpleasant lighting pattern. To make sure that the light is distributed evenly over the areas, a evenness measurement is available. The following formula is used

$$fit_{unit} = L_{min}/L_{avg}$$

where L_{min} is the measurement point with the smallest value, and L_{avg} is the average of all measurement points. This gives a percentage to which the system will try to maximize. The effect of the equation is the even distribution of the light in the room, such that the lowest light reading the as close to the average as possible, stopping extremes.

4.4.3 Glare

Glare can be measured in many different ways. For these experiments, the Visual Comfort Probability (VCP)[20], also known as the Guth Visual Comfort Probability, is used. It is a metric used to predict the percentage of persons who will find a certain scene or space visually comfortable. Radiance contains a system to perform this measurement. A rendered scene is given to Radiance, and visual direction points are also given. The glare is measured at these angles from the view point, and the VCP is calculated. This percentage is given back to the GP system for use in fitness.

4.4.4 Colour

For experiments that are concerned with colour, a target colour map is used. Nine positions are used to gather rtrace information, and these colour channel values are then compared to the target using a sum of errors of a ratio comparison between channels. This is done to work around the domain issue of the standard RGB colour space being compared with an unbounded HDR colour space.

$$Value1 = RedChannel \div GreenChannel \tag{4.1}$$

$$Value2 = RedChannel \div BlueChannel \tag{4.2}$$

$$Value3 = GreenChannel \div BlueChannel \tag{4.3}$$

Chapter 5

Basic Experiments

This chapter describes some basic experiments. It examines illumination with artificial lighting, the use of skylights for natural lighting, and day and night illumination measurements.

5.1 Common Parameters

For the basic experiments conducted, many of the GP parameters are shared between them (Table 5.1). Detailed explanation of parameters are given by [23]. These parameters represent the basic GP set-up, which has not been finely tuned for optimal performance on the problems, but selected for acceptable runtime performance and reasonable tree size creation. By keeping parameters the same in each experiment, the change in independent variables becomes more apparent. If any changes are made to the parameters, they are reported in the corresponding section.

The modelling environment used in the experiments has been created to be a simple test space where the number of secondary factors to the lighting analysis are minimized. The room consists of a plain square box, with sides which simulate a painted drywall surface. The tables and chairs have been placed in the room to show the effects of the resulting lighting, creating shadows and illuminating the different surfaces. Table 5.2 lists the Radiance-based parameters that are being used in the room.

Parameter	Value
Runs	10
Generations	50
Population Size	250
Initialization Method	Half-and-Half
Max Tree Depth	11
Grow Tree Max Depth	6
Grow Tree Min Depth	4
Full Tree Max Depth	6
Full Tree Min Depth	4
Tournament Size	3
Crossover Rate	90%
Mutation Rate	10%
Mutation Grow Tree Max Depth	4
Mutation Grow Tree Min Depth	2

Table 5.1: Common GP Parameters for Basic Experiments

Parameter	Value
Room Width	20m
Room Length	20m
Ceiling Height	3m
Wall Material	Plastic(0.6,0.6,0.6,0,0)
Ceiling Material	Plastic(0.8,0.8,0.8,0,0)
Floor Material	Plastic(0.3,0.3,0.3,0,0)
Glass Definition	Glass(0.69,0.69,0.69)
Light Size	0.125m

Table 5.2: Common Radiance parameters for basic experiments

5.2 Brightest Room

5.2.1 Setup

The first and most basic experiment is an unbounded search for a maximum lux level. This is accomplished by allowing the system to create as many light objects as possible. Each light object is created with a pre-set light intensity, without restriction of placement. This can result in the overlapping of the light objects. Illumination levels are measured at sixteen evenly spaced points, one meter above the ground. The sum total of the illuminance values are used as the raw fitness value.

The functions listed in Table 5.3 are used as the base for these experiment.

Type	Function Name
R	`Root(LM)`
LM	`Top_Light(LB)`
LB	`Light_Filler(LB,LB)`
LB	`Basic_Light(F,F)`
F	`ERCFloat`

Table 5.3: Base GP language

- Top_Light(LB) — The top level of the light object creation tree. Allows for the creation of a large tree where leafs create a single light object.
- Light_Filler(LB,LB) — A filler function to expand the size of the tree. Used to try to create a full tree of maximum depth.
- Basic_Light(F,F) — The function which creates a light object in the environment. Takes in two floats, which denote the x and y position of the object.
- ERCFloat — A random float type to give x and y values to the Basic_Light function.

5.2.2 Results

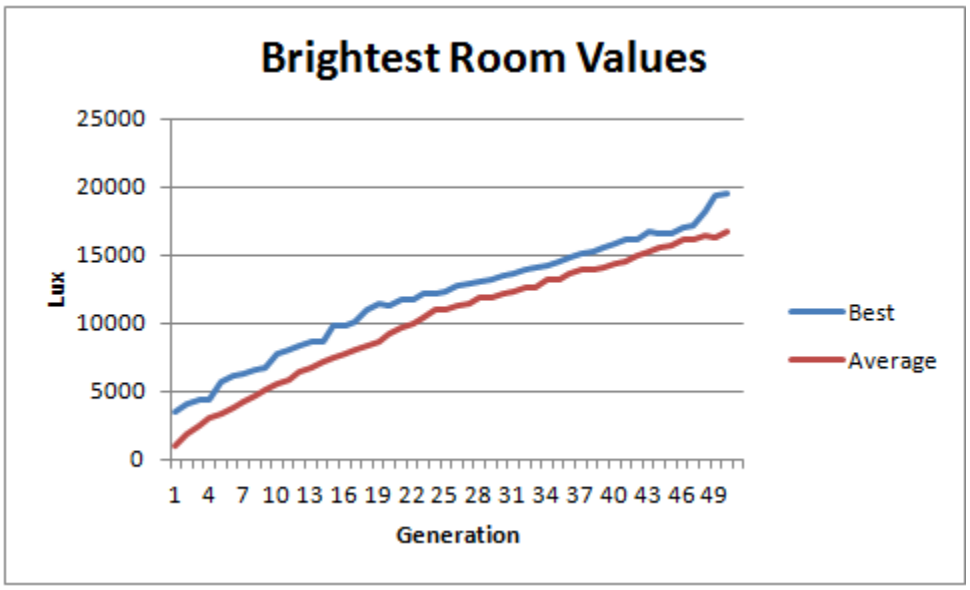

Figure 5.1: The population best and average over 50 generations(average over 10 runs)

As shown in Figure 5.1, there is no convergence in the generations shown to be taking place, and the illuminance reading is limited only by the size of the GP tree which created it. The fuller and deeper the tree, the more light objects it would be able to create. The light objects had a cumulative effect. An example of how bright the room could get is shown in Figure 5.2.

Figure 5.2: The best found solution maximizing illuminance

5.3 Controlled Spacing

5.3.1 Setup

The previous experiment relied in the stacking of lights, which created unrealistic solutions which would not be physically possible to recreate in a real environment. To refine the creation process, a way to stop lighting overlap is required. Spacing the lights a minimum distance from the walls and each other allows the problem to be tried again with more realistic results. Upon evaluation of the GP tree, each light object is first checked against a list of lights which had already been created in the room. If the light to be created has a coordinate within three metres of a light which was successfully created, or within two metres of a wall, the creation will not take

place. This limits the number and density of lights which can be created, and forces the GP to seek the optimal fit of lights within the given constraints. There is no penalty set for this action in fitness evaluation, and therefore bloat could happen within the GP tree, as the non-placed lights would be neutral to the score. The same fitness evaluation of sixteen illumination measurement locations in the room is used as before.

5.3.2 Results

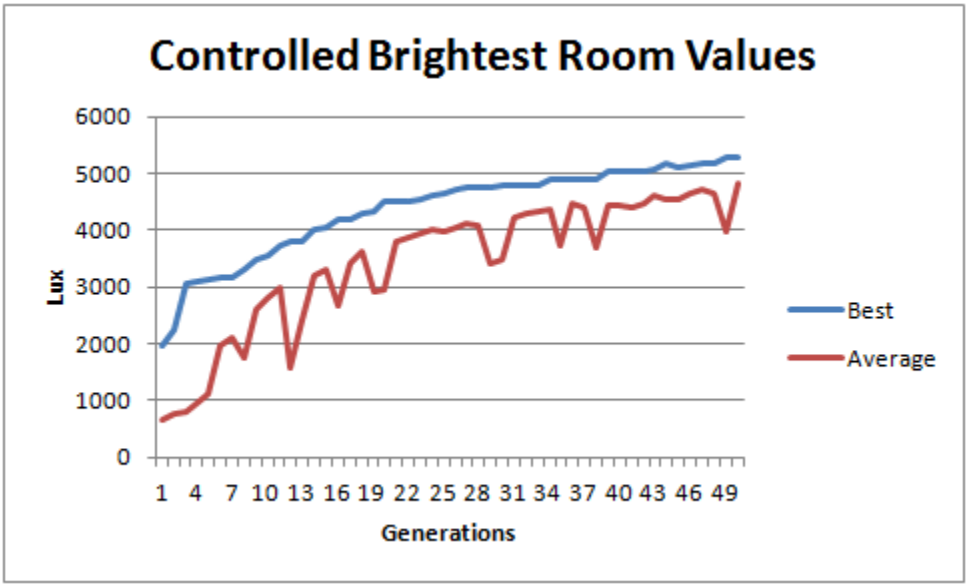

Figure 5.3: The population best and average over 50 generations(average over 10 runs)

As Figure 5.3 shows, in relation to the non-restricted experiment before, the fitness rises steadily and without convergence. The average population had many results which were penalized, resulting in many dips in fitness. This shows that the one dimensional problem of maximum illuminance is a relatively easy problem for GP to solve, and means that we can add more dimensions to the problem. The scene in Figure 5.4 shows the best found solution to this problem.

Figure 5.4: The best found solution controlling spacing

5.4 Skylights

5.4.1 Setup

Next, the introduction of natural light is considered. To do so, the option of a sky light is introduced. Light objects are still constricted to the controlled spacing of before, as well as being restricted from being placed on the skylight. The maximum size of the skylight is restrained to a 10m $\times$ 10m square, as preliminary testing showed the entire ceiling being taken over completely by the window.

A new objective added is the even distribution (or evenness) of the lighting. The goal is the placement of artificial lights evenly around the outside of the room over each table, as well as the creation of a large, square skylight in the centre of the room. This type of skylight would work best at distributing the sun's rays around the room. Noon was used to maximize the overhead light from the sun. Different times of day would result in differing solutions. The new objective is evenness (see Section 4.4.2). These objectives together work to maximize the light exposure at the target points. As in previous experiments, a set of points positioned one metre above the ground are designated as measurement points for the illumination values. A total of eight

points are used, each located directly on top of each of the tables of the outer rim. This was chosen to keep the level of direct influence from the sun smaller, as the high illumination potential of the sun would overburden all other points. This experiment was done using the normalized sum of ranks method for fitness evaluation, using two objectives: evenness of the light, and illumination level.

Type	Function Name
R	`Root(C,LM)`
C	`SkyLight(F,F,F,F)`

Table 5.4: Skylight GP changes

Changes to the GP language for this experiment are given in Table 5.4. These new functions are:

- Root(C,LM) — The root function was modified to accommodate the addition of the skylight function. The creation of light objects and the sky light are kept separate.
- SkyLight(F,F,F,F) — The Skylight creation function takes in 4 floats. The floats are paired into two x and y coordinates. These pairs represent two opposite corners of the rectangular skylight. A polygon is created with a hole using these coordinates, and a glass panel is placed in the space.

5.4.2 Results

The rise in lux readings in Figure 5.5 are indicative of two targets being met. The first is the maximum allowed size of the skylight, generally staying as close as possible. The second is the placement of the artificial lights. The higher the illuminance reading from the targets, the closer the lights are placed directly on top, creating the maximum level of illumination. This shows as well in the distribution of light. Solutions closer to the target would have a more even distribution than others.

Figure 5.6 shows the rise in the evenness of the distribution rises at almost the same rate as the illuminance reading. This is showing that the level of illumination in the rooms is reaching a peak. The light from the sky light is spread as far as it can, and the remaining importance is the placement of the artificial lights. The slower climb in this experiment compared to the other shows the more fine tuning that this problem needs over the others, as there is an optimal for the GP to reach. This optimal would be a maximum sized sky light with one light directly located over

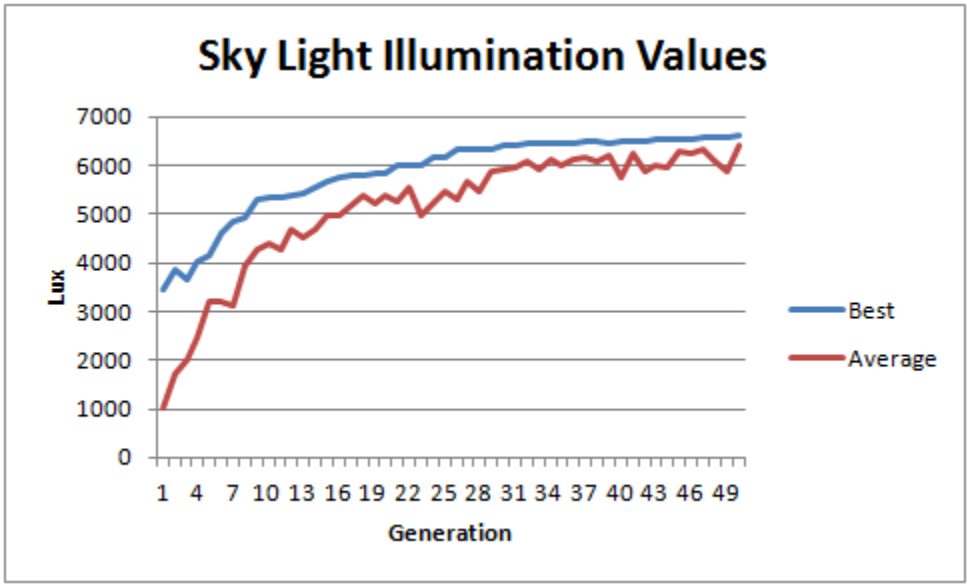

Figure 5.5: The population best and average over 50 generations(average over 10 runs)

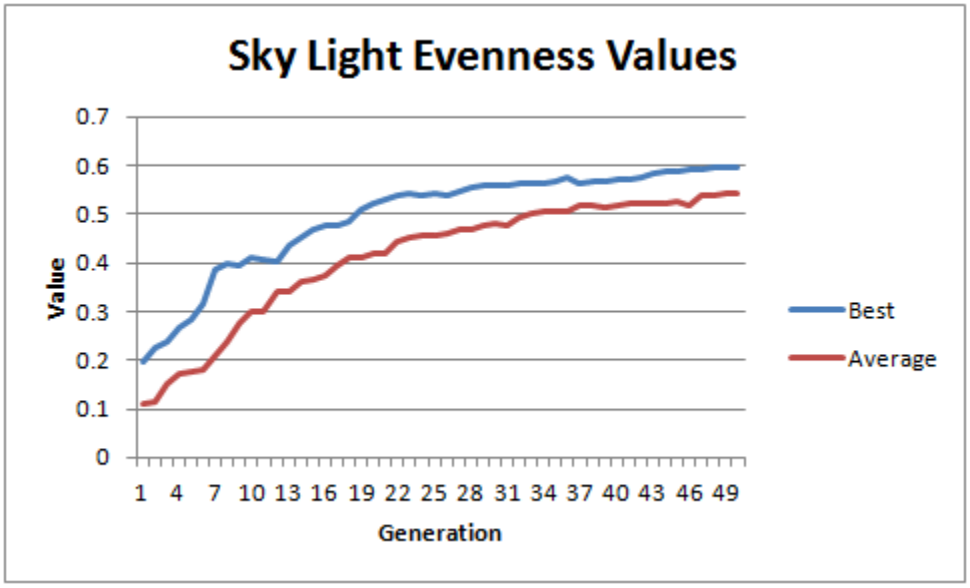

Figure 5.6: The population best and average over 50 generations(average over 10 runs)

each of the target points. The scene in Figure 5.9 (a) shows how the sky light changes the layout of the artificial lights.

5.5 Night and Day

5.5.1 Setup

The inclusion of the sun and skylight into the calculation opens a new kind of problem, which is the difference between daytime and night time illumination. A room which satisfies daytime requirements may not be acceptable at night. Each solution found is to be tested with the same two objectives as before: maximizing lux value and even distribution of light. The change would be that the measurements would be taken twice. One measurement is to be taken at high noon with full sunlight shining, and the second measurement is to be taken on a moonless night. This allows the system to find solutions which have a balance of artificial lights for the night while taking advantage of passive solar illumination during the day.

5.5.2 Results

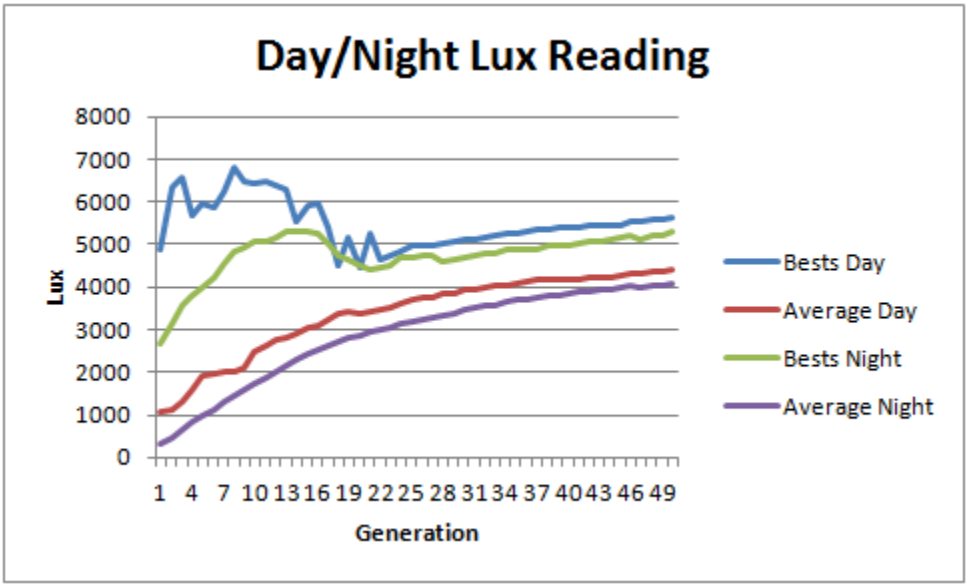

Figure 5.7: The population best and average over 50 generations(average over 10 runs)

The result graphs shown in Figure 5.7 and Figure 5.8 show interaction between the four objectives. The high value of the day time reading is a direct result of the sky light allowing for a high level of sunlight into the room. This has a detrimental effect of skewing the evenness of the lighting during the day. The area under the skylight causes the total average to increase, increasing the difference between the

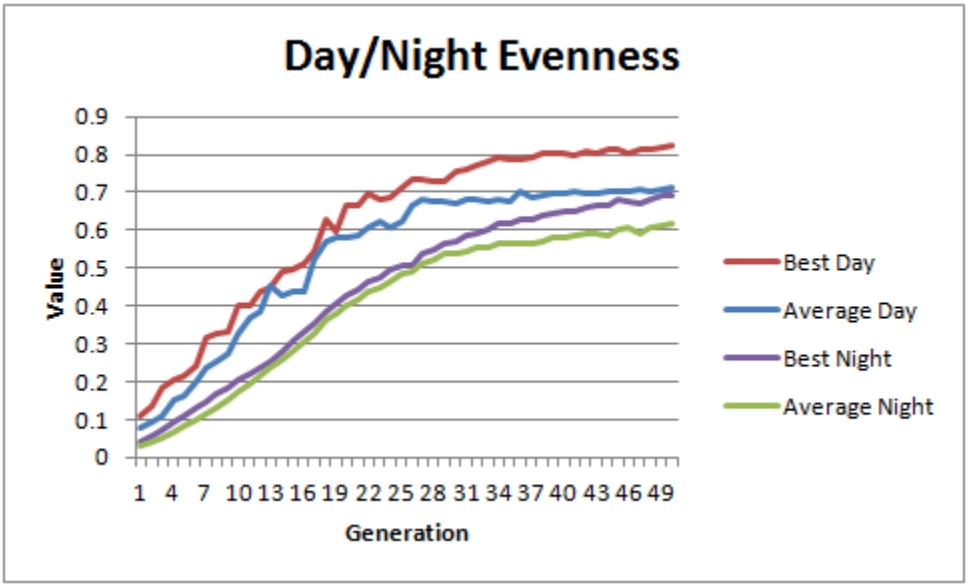

Figure 5.8: The population best and average over 50 generations(average over 10 runs)

highest and lowest reading. This is dealt with by evolution through shrinking the skylight, and placing more emphasis on artificial lighting. This causes the evenness to rise, while bringing down the total reading, requiring more lights to be added. The night reading also has a similar rise-fall-rise pattern. This can be contributed as well to the relation between the maximization of the lights and evenness. More lights are added during evolution, which will raise the one objective, but the lights have yet to be moved to a more uniform distribution. All four of these objectives eventually come in line and work together to simultaneously improve.

The scene in Figure 5.9 (b) shows how the sky light was sized down during evolution. Figure 5.9 compares this solution with the one from Section 5.4, showing the effect of the night based evolution on the room.

5.6 Summary

This chapter presented experiments focusing on the basics of lighting. The two types of lighting, artificial and natural (skylight), were shown alone with their effects and strengths. Using multiple objectives, the first step towards relating artificial lights, natural lighting, and evenness was done.

(a) Example of best solution with a skylight.

(b) Example of best solution with night and day cycle.

Figure 5.9: Sky light comparison

Chapter 6

Window Experiments

This chapter details the experiments which were done to add more complexity to the problem, and to show in more detail the relation between the different possible objectives and applications. The addition of windows and materials adds a great deal of new possibilities, giving more control over a number of aspects of the illumination problem. This moves the system from mostly user defined, to a more automated design system. Previously, almost all aspects of the system were pre-set. With each succeeding experiment in this chapter, more attributes of the room are given to automation.

6.1 Common Parameters

Table 6.1 lists the common GP parameters for these experiments. The number of runs are increased from previously to gain a better statistical sample.

Table 6.1 contains the basic Radiance parameters for these experiments. Unless otherwise noted, these are the values used. To make the results appear more realistic, the scene is designed as follows. The walls are painted red, and a hardwood floor is used. The sun is set at noon on an overcast day.

The functions listed in Table 6.3 are used as the base for these experiments. Any changes are noted in the respective section.

Parameter	Value
Runs	20
Generations	50
Population Size	150
Initialization Method	Half-and-Half
Max Tree Depth	11
Grow Tree Max Depth	6
Grow Tree Min Depth	4
Full Tree Max Depth	6
Full Tree Min Depth	4
Tournament Size	3
Crossover Rate	90%
Mutation Rate	10%
Mutation Grow Tree Max Depth	4
Mutation Grow Tree Min Depth	2

Table 6.1: Common GP Parameters for Experiments

Parameter	Value
Room Width	20m
Room Length	60m
Ceiling Height	6m
Wall Material	Plastic(0.390,0.051,0.051,0,0)
Ceiling Material	Plastic(0.8,0.8,0.8,0,0)
Floor Material	Radiance library oak floor
Glass Definition	Glass(0.96,0.96,0.96)
Light Size	0.125m
Sun Parameters	12/4 12:00 -c 55.86°N 0°E

Table 6.2: Common Radiance parameters for basic experiments

6.2 Window Creation

Here, two different window creation techniques are introduced, as well as the combination of windows, skylights, and artificial and natural lighting.

6.2.1 Setup

Window Technique 1

The following experiment supplements the base GP language as noted in Table 6.4. The window function "$North\_Wall(I, F)$" contains only two parameters, rather than

Type	Function Name
R	`Root(W,LM)`
LM	`Top_Light(LB[n..m])`
LB	`Basic_Light(F,F)`
W	`Windows(C,NW,SW,EW,NW)`
C	`SkyLight(F,F,F,F)`
NW	`North_Wall(I,F,F,F,TF)`
SW	`South_Wall(I,F,F,F,TF)`
EW	`East_Wall(I,F,F,F,TF)`
WW	`West_Wall(I,F,F,F,TF)`
TF	Add(TF[2..4])
TF	-, *, /, neg, sin, cos, log
TF	X, Y
TF	`ERCTF`
F	`ERCFloat`
I	`ERCInt`

Table 6.3: Base GP language

the four of the base language. This function uses a modulo integer expression to set the number of windows on a wall, and ranges from 0 to 30. This number determines the width of the windows, making an equally sized repeating pattern of window and wall. A floating point value is used to set the percentage of the wall's vertical space it uses. It ranges from 0% to 100%.

Type	Function Name
NW	`North_Wall(I,F)`
SW	`South_Wall(I,F)`
EW	`East_Wall(I,F)`
WW	`West_Wall(I,F)`

Table 6.4: Window 1 GP language

Fitness evaluation is divided into six separate objectives. The room is split into three 20m × 20m sections. Sixteen points are evenly spaced in a 4×4 grid to be points of measurement in each section. Each section defines an objective as follows:

1. The south section is allowed to grow in brightness as much as possible.

2. The middle section is measured based on the absolute difference between its average measurement and half the average of the brightest area.

3. The north section is measured as the absolute difference between its average measurement and half the average measurement of the middle region.

Three objectives measuring the even distribution of the light in each section are also used. This gives a total of six objectives. This controls the placement of the skylight and the distribution of the windows and artificial lights with a greater effect than one measurement across the entire area.

Window Technique 2

A second method for window creation allows for more flexibility in the patterns created which might satisfy the requirements of the room better. It also allows for more interesting styles of windows, ranging from wall spanning windows to small port hole windows. Using the same method of determining panel width as technique 1, this method of creation does not evenly pattern the windows and wall panels as before. The size and location of the window on the section of wall is determined by values given by the GP system. This allows for greater control over the aperture of the walls as well as the positioning of where the light enters. The function "$North\_Wall(I, A, B, C, TF)$" is used, where A, B, and C are mantissas where $0 \leq F \leq 1$. TF is a GP tree function which uses the position of the window to determine if the window will be drawn in that space or a blank wall piece will be used. Figure 6.1 shows the graphical representation of how the window is created.

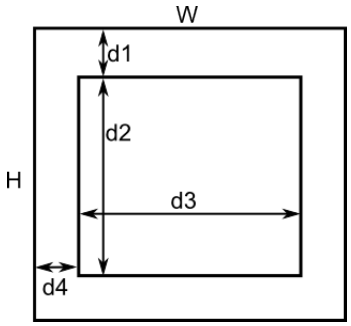

Figure 6.1: Window creation scaling

Consider the following equations:

$$d_1 = a\% \times H \tag{6.1}$$

$$d_2 = b\% \times (H - d_1) \tag{6.2}$$

$$d_3 = c\% \times W \tag{6.3}$$

$$d_4 = \frac{(W - d_3)}{2} \tag{6.4}$$

Here, d_1 represents the height of the top of the window, as a percentage of the height of the wall. The value of d_2 determines the total height of the window itself, returning a percentage value of the remaining height of the wall. Any space between the bottom of the window and the floor is used as wall. The width of the window is determined by d_3, with d_4 being used to center the window in the panel area.

The same 6 objectives used for technique 1 are used for technique 2.

6.2.2 Results

Window Technique 1

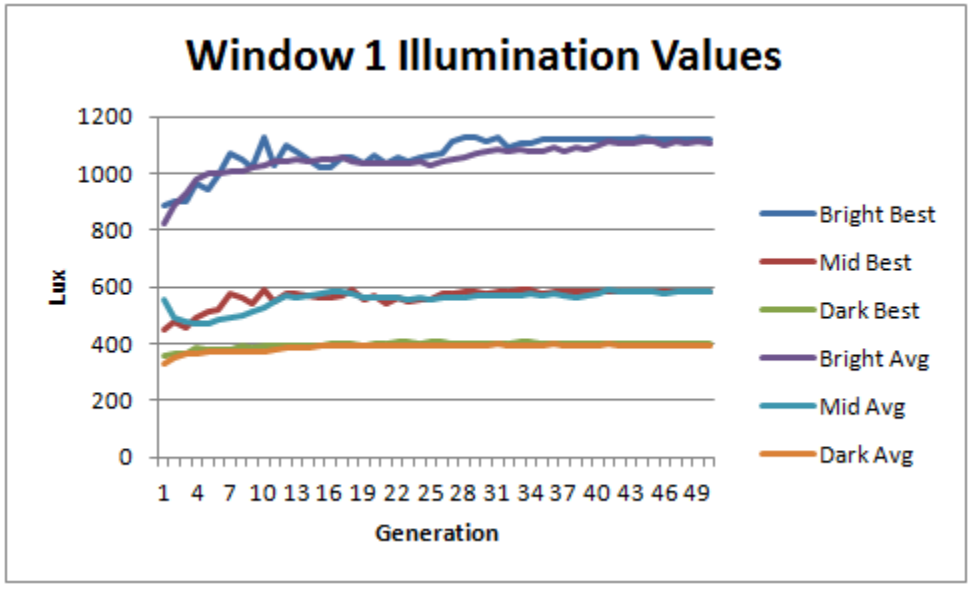

Figure 6.2: The population best and average over 50 generations avg. 20 runs

The results shown in Figure 6.2 and Figure 6.3 show that the population for this window technique begins to converge quickly and then slow down for the rest of the run. The biggest determiner of convergence is the south most section designated

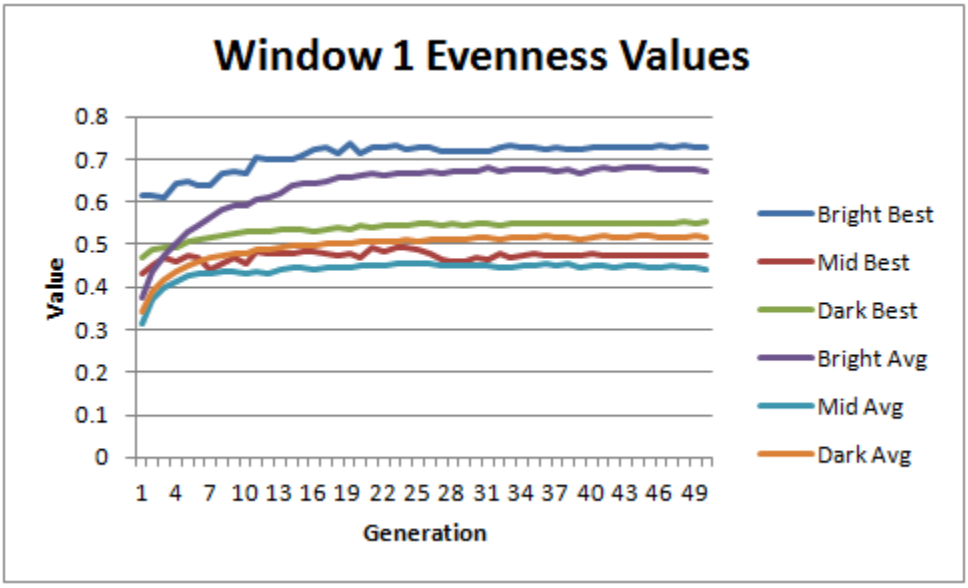

Figure 6.3: The population best and average over 50 generations avg. 20 runs

to become as bright as possible. This objective strongly influences the two other sections.

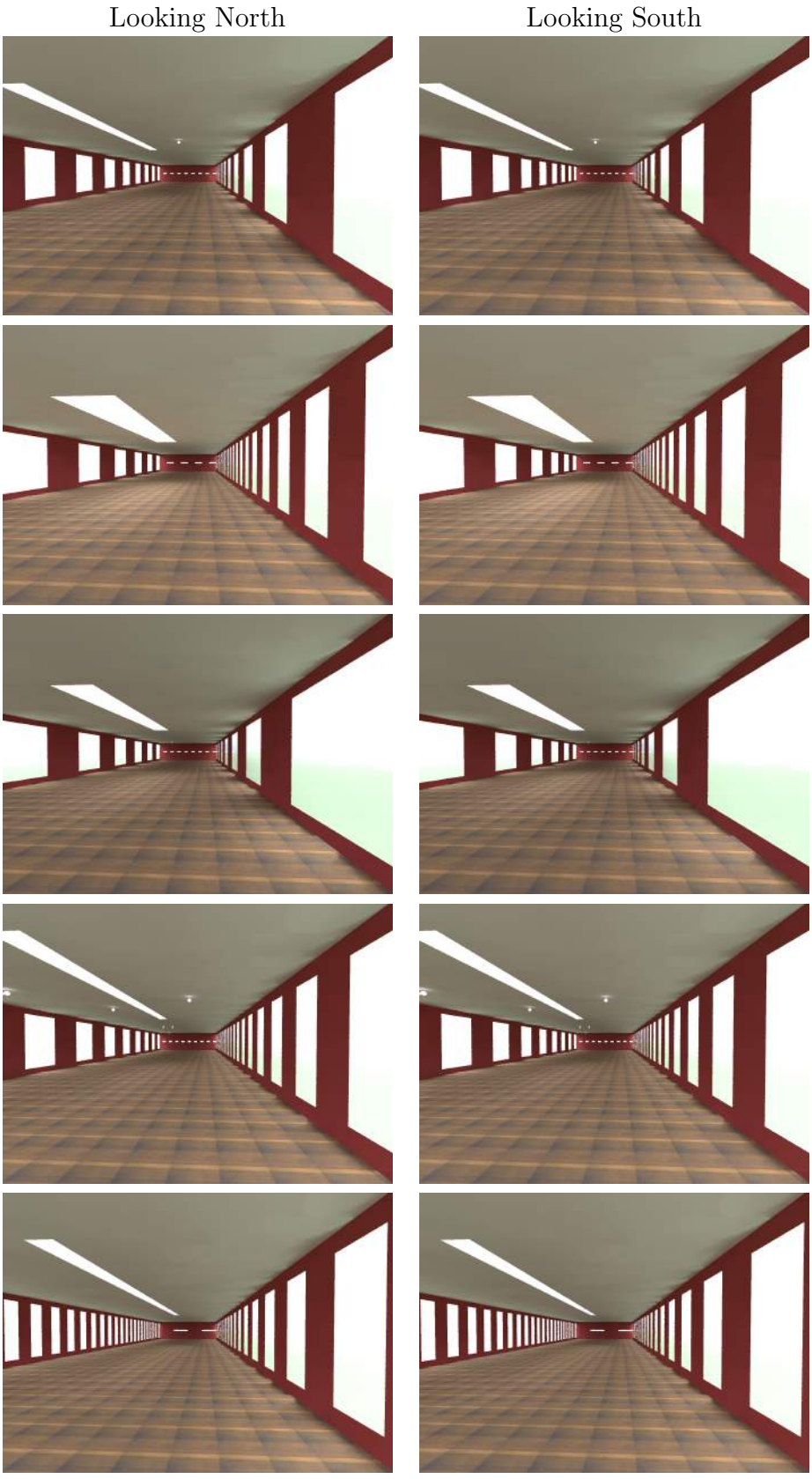

Figure 6.4: First window creation method

The images shown in Figure 6.4 show the top 5 ranked solutions generated using this technique. They show that the use of artificial lights was kept to a minimum, relying almost entirely on the use of passive solar illumination. The small windows at the northern end of the room are a result of the system trying to darken that end of the hall. Notice consistences in skylights being long and thin, and window design being high in number as well as tall and thin.

Window Technique 2

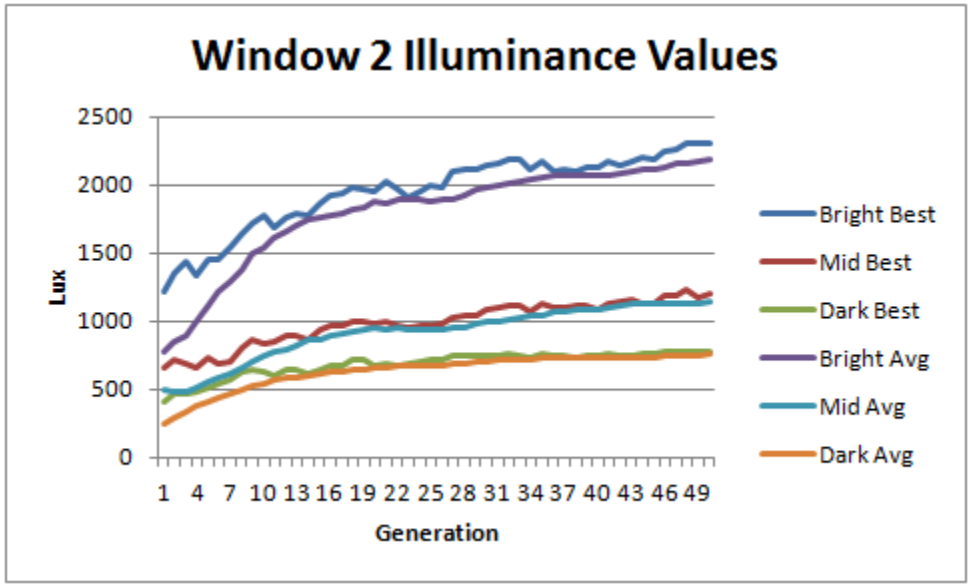

Figure 6.5: The population best and average over 50 generations avg. 20 runs

Unlike the first window technique, the convergence of this technique happens much later in evolution, if at all, as shown in Figure 6.5 and Figure 6.6. It is also able to rise to greater values for the maximum brightness of the first objective. This can be attributed to the greater flexibility of the non-window space on the walls. This window creation technique allows the system to maximize the window space to let more passive solar illumination into the room.

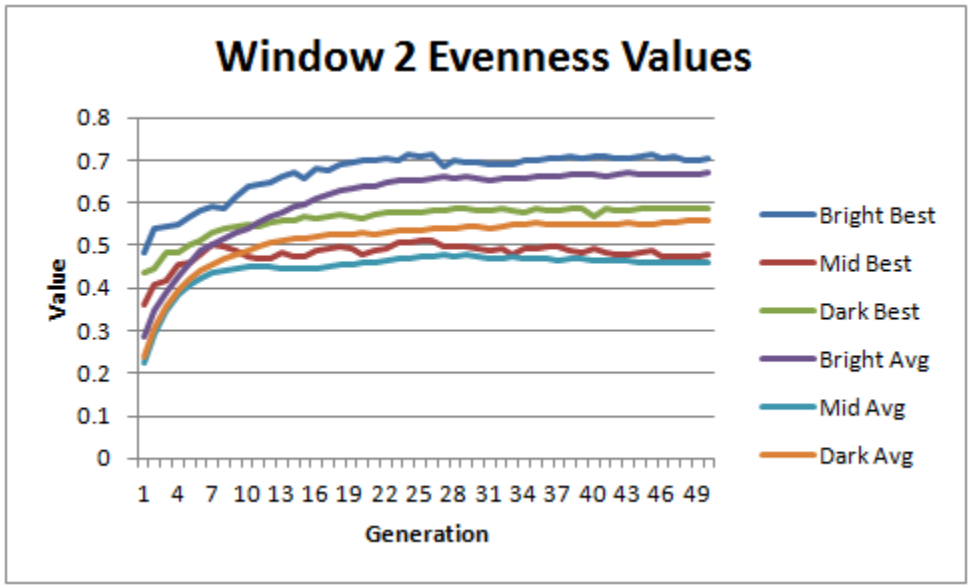

Figure 6.6: The population best and average over 50 generations avg. 20 runs

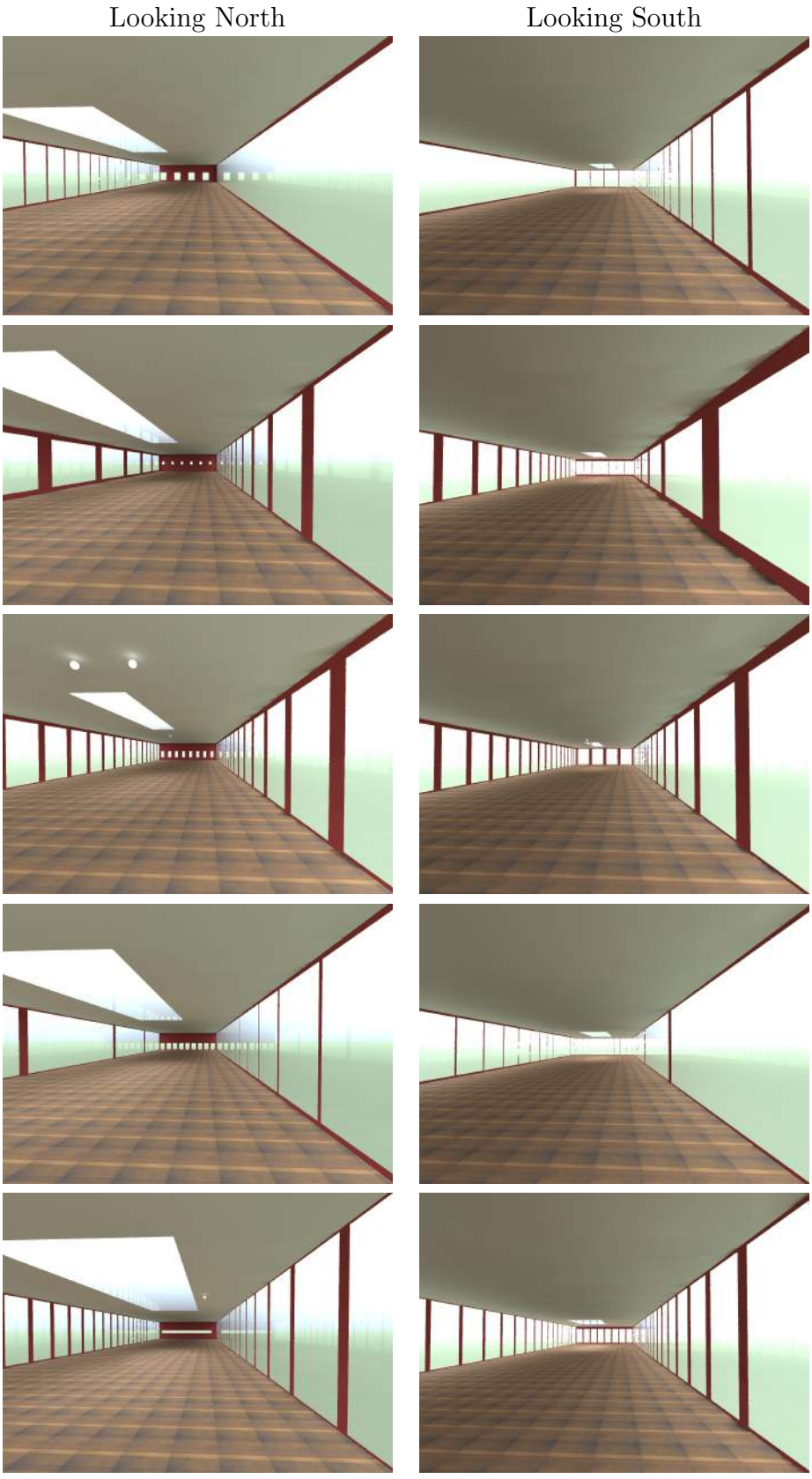

Figure 6.7: Second window creation method

We can see in Figure 6.7 that the variation of windows created from this technique is far greater than the first. The greatest difference can be seen between a very large, single window filling an entire wall and the smaller porthole windows seen at the north of each room. This technique has been shown to generate better quality solution than the uniform distribution of windows in technique 1.

6.3 Night and Day

6.3.1 Setup

Here we add night measurements to find if the system is able to make a more generally useful room which can be used any time of day. Previously we had seen that the room would be created with less focus on artificial lighting, using the power of the sun to fulfil its requirements. To test the window creation with respect to night time measurement, a different measurement is required for the three separate areas. Using UDI measurement values[29] as an inspiration, the fitness is changed to have a more definite target. Lights are turned off during the day, and back on during the night. The first objective is changed to the absolute difference between the average measurement and a value of 4000, which is double the maximum value in the acceptable range of UDI. The other two sections are measured in the same way, with a target of 1000, an acceptable UDI value for a work area, and 0 to create an extreme difference between the two ends and force more noticeable results. The light distribution objectives are combined into one sum of measurements. Night time measurements are taken at two sections. The first is the south most section. During the day this is the brightest section of the room, but at night it is given at target value of 0. The opposite end of the room is given a night time target of 500. This results in a total of 6 objectives.

6.3.2 Results

The results shown in Figures 6.8, 6.9, and 6.10 show the system beginning to struggle with the added complexity of the problem. The value of the brightest section of the day fluctuates as it tries to optimize the other objectives. This is the give and take of the sum of ranks that is commonly seen.

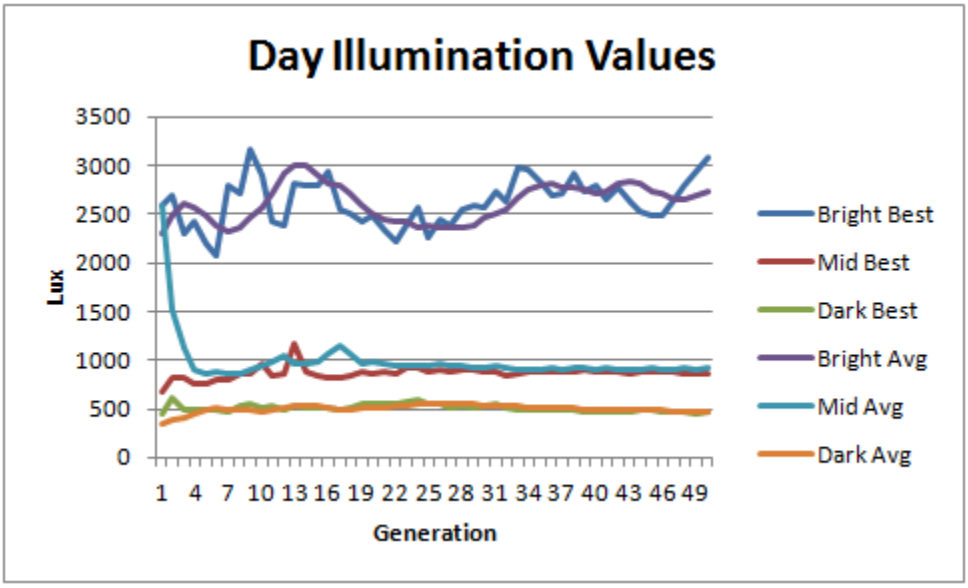

Figure 6.8: The population best and average over 50 generations avg. over 20 runs

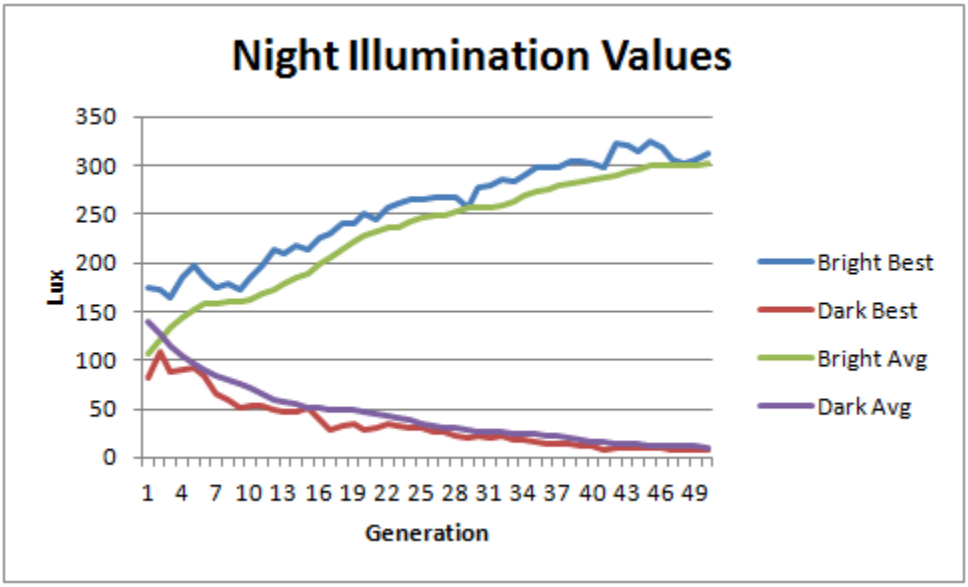

Figure 6.9: The population best and average over 50 generations avg. over 20 runs

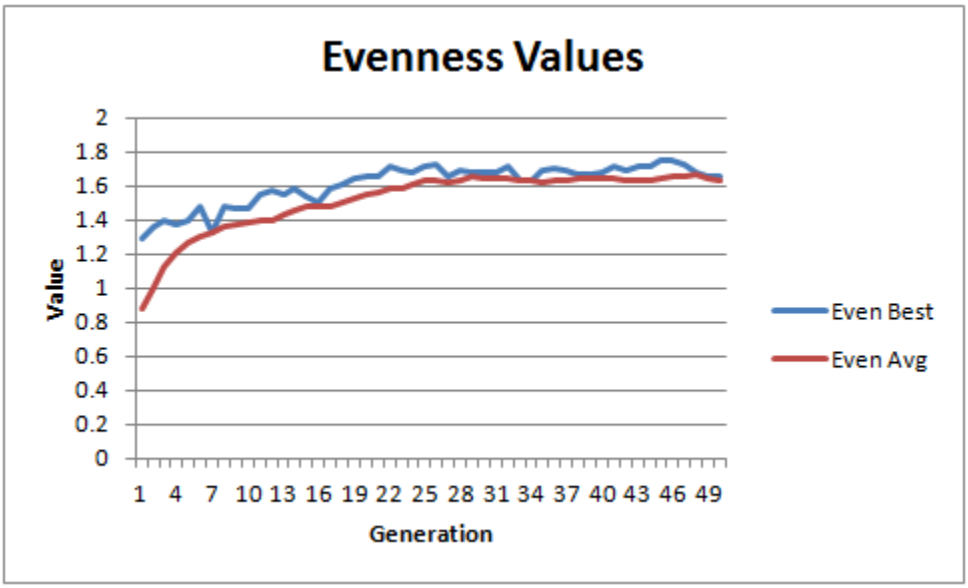

Figure 6.10: The population best and average over 50 generations avg. over 20 runs

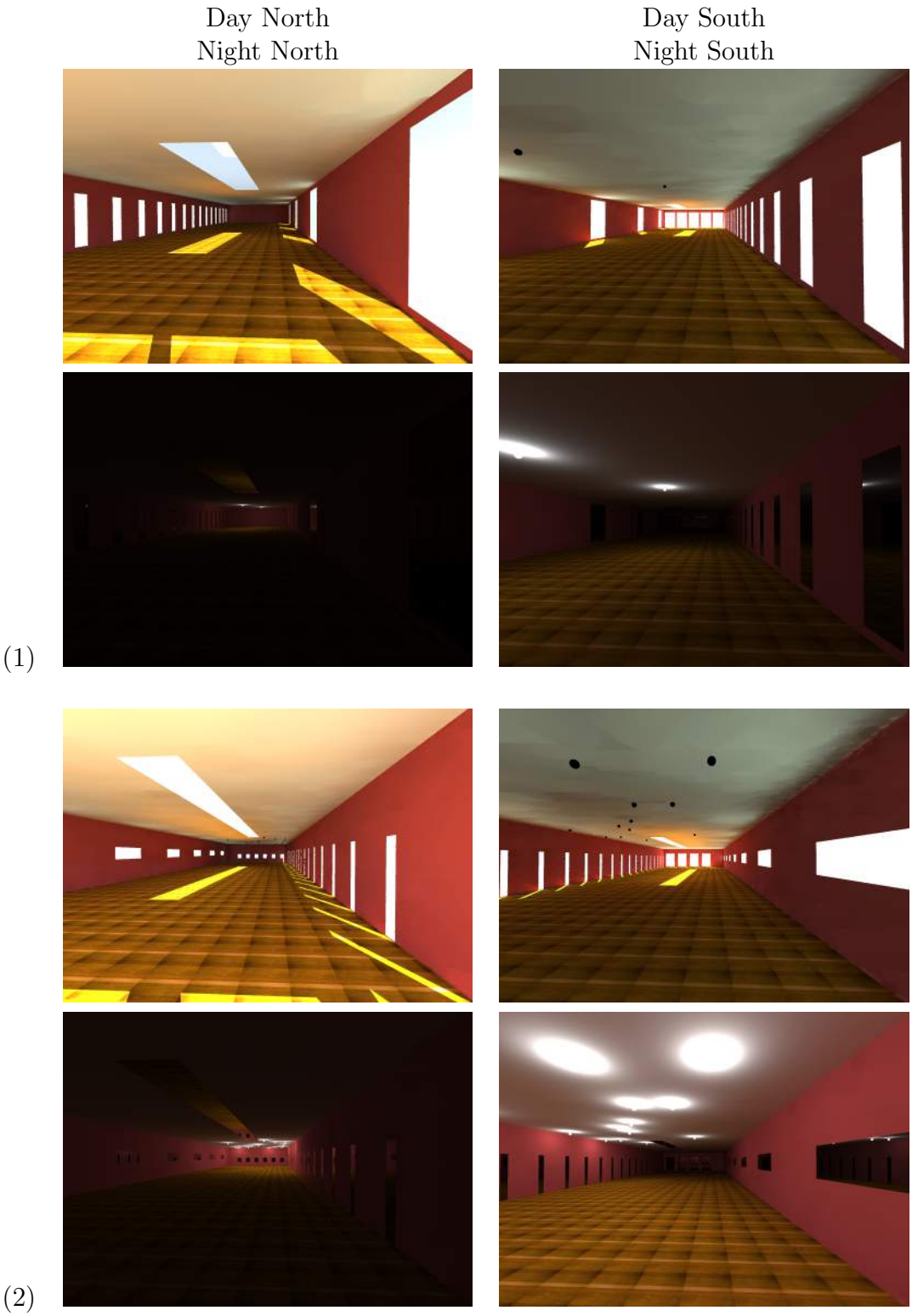

Figure 6.11: Day and Night examples

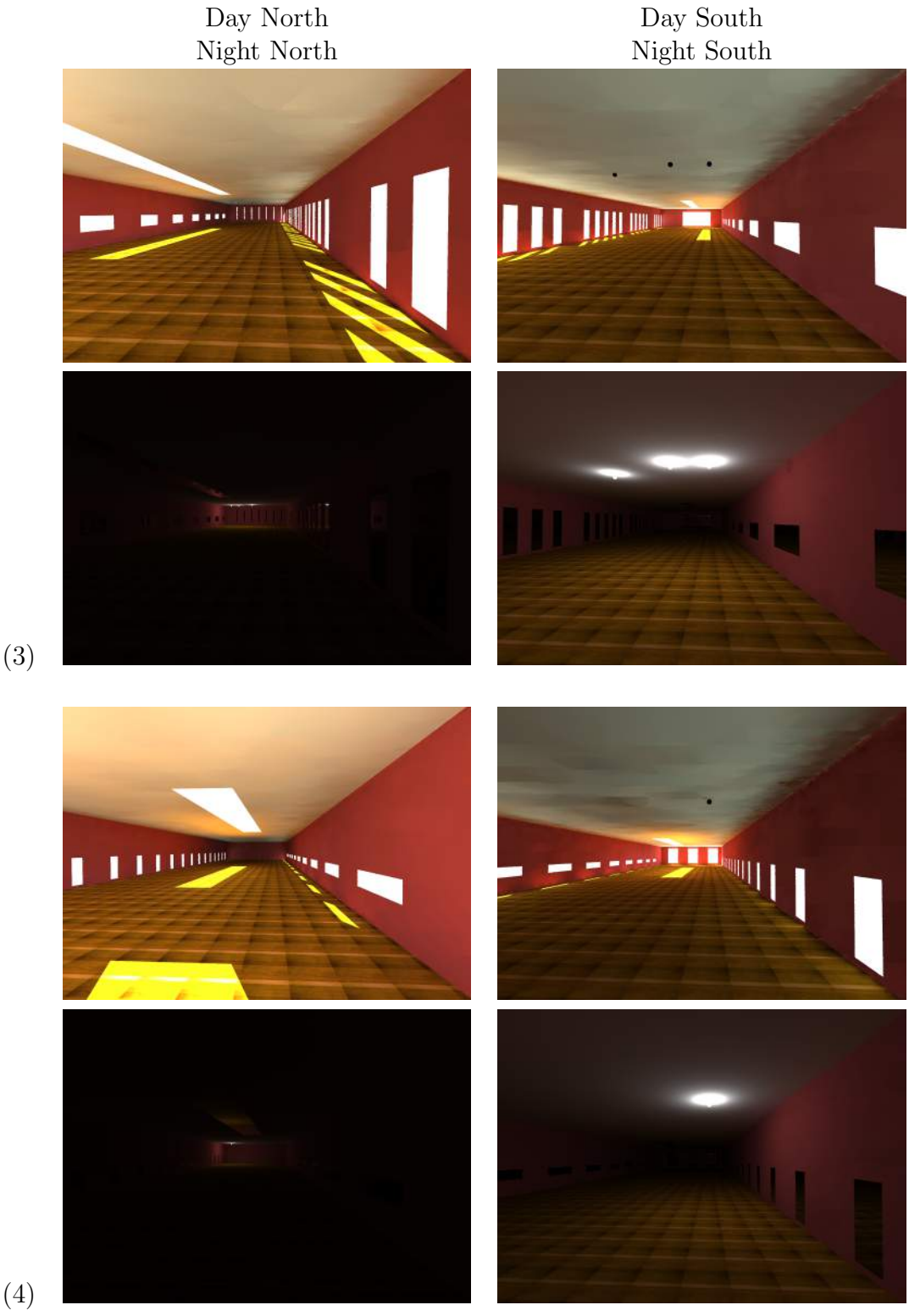

Figure 6.12: Day and Night examples cont.

Figure 6.13: Day and Night examples cont.

As shown in Figures 6.11, 6.12, and 6.13, the results are very different than before. The windows used are much smaller, thanks to the higher constraints on the objectives. As before, north side windows are kept very small. Sky lights are used with the windows to try to maximize the light in the bright side section. The night time objectives were able to be achieved easily with the artificial lighting. The bright day objective was the most volatile while the system tried to balance the 6 objectives. This is caused by the adjustments of the side window sizes. The bright objective would be the most beneficially affected by larger windows, so the system would be trying to find a balanced size which does not overpower the other two day objectives. An example of the GP tree generated from this experiment can be found in Appendix A.

6.4 Material Definition

In these experiments, the GP is allowed to evolve material definitions for walls, floor, and ceiling. The measure of its effect on illumination is used, as well as the measure-

ment and optimization of glare.

6.4.1 Setup

The following changes shown in Table 6.5 were made to the base GP language for this experiment.

Type	Function Name
R	`Root(W,MM,LM)`
NM	`North_Material(M)`
SM	`South_Material(M)`
EM	`East_Material(M)`
WM	`West_Material(M)`
CW	`Ceiling_Material(M)`
FW	`Floor_Material(M)`
M	`Material(F,F,F,F,F)`

Table 6.5: Material GP language

Radiance allows control over the definition of all materials in a scene. For the three basic materials that could be used for the walls of a room, five factors are available for use: red, green, and blue colour channels, specularity, and roughness. Specularity controls the reflective property of the material, from a matte material increasing to satin. Roughness values define the amount of polish on a surface. A higher roughness causes a scatter of the light, giving a granular look. For the control of the extreme measures which can arise, the roughness parameter was set to a zero value for all experiments. The reasoning behind allowing the system to have control over the material definition is to see the effects of colour and reflection on illumination. The colour aspect would have less of an effect because the measurements are being taken and converted to white light, but having different colour possibilities might make for interesting design potentials.

Three materials are defined for selection: plastic, metal, and mirror. The first two materials work similarly and share the same parameter definition. The difference between the two is the metal material type highlights are affected by the material colour. The mirror material uses only the red, green, and blue colour channels. These channels define the red, green, and blue reflectance values.

This first experiment with material uses an unconstrained material definition. Using the same parameters as Section 6.3, the direct effect of materials would be

measurable and comparable to the data which had been collected before. The same objectives and parameters are used as before in Section 6.3, with 6 objectives.

6.4.2 Results

As can be seen in Figures 6.14, 6.15, and 6.16, a high amount of glare is present. Many solutions used a high amount of reflectivity for the materials, relying less on the large or numerous windows seen in previous experiments. The reflective surfaces were able to spread the lighting over a larger range.

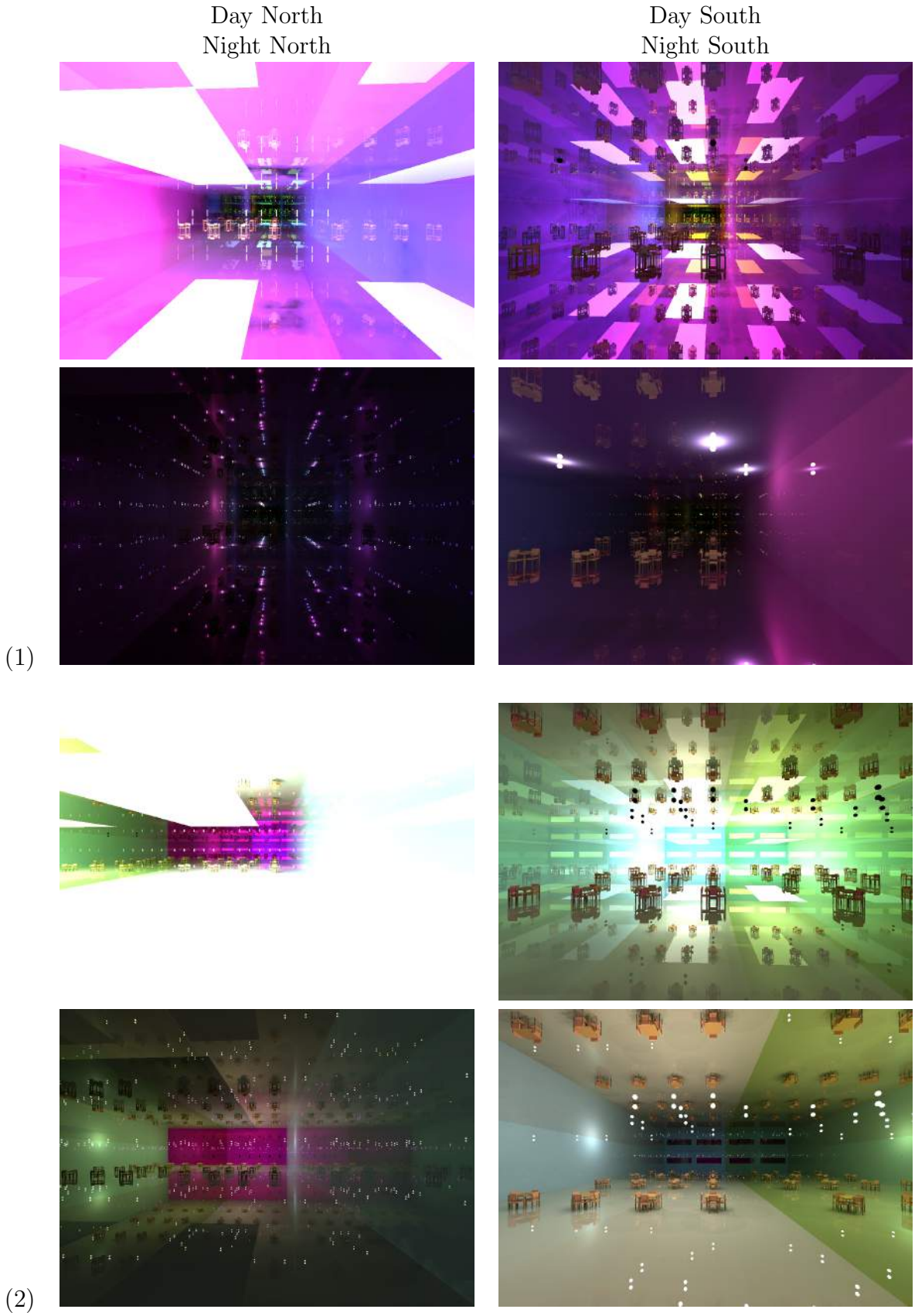

Figure 6.14: Top 5 material generation results

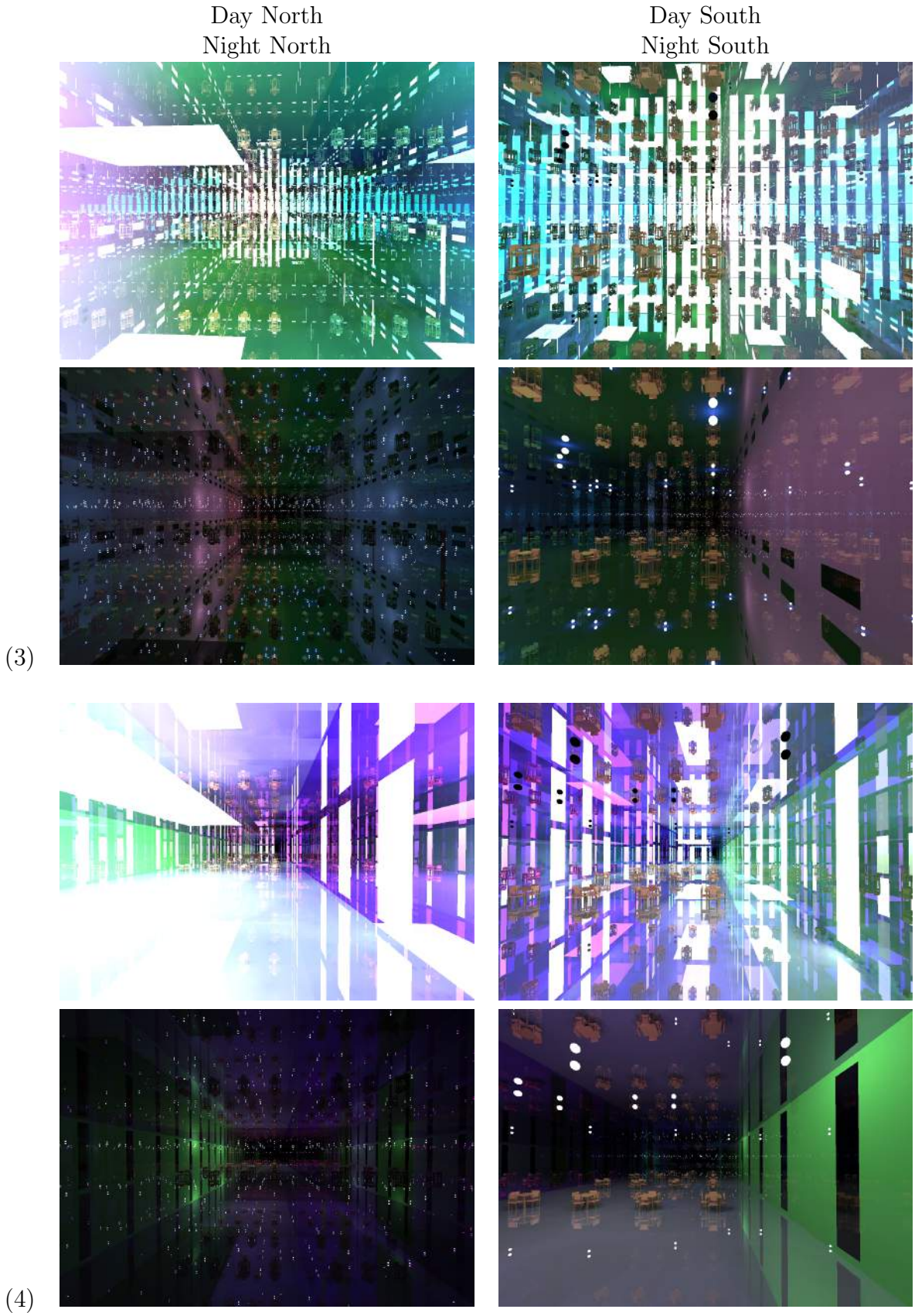

Figure 6.15: Top 5 material generation results cont.

(5)

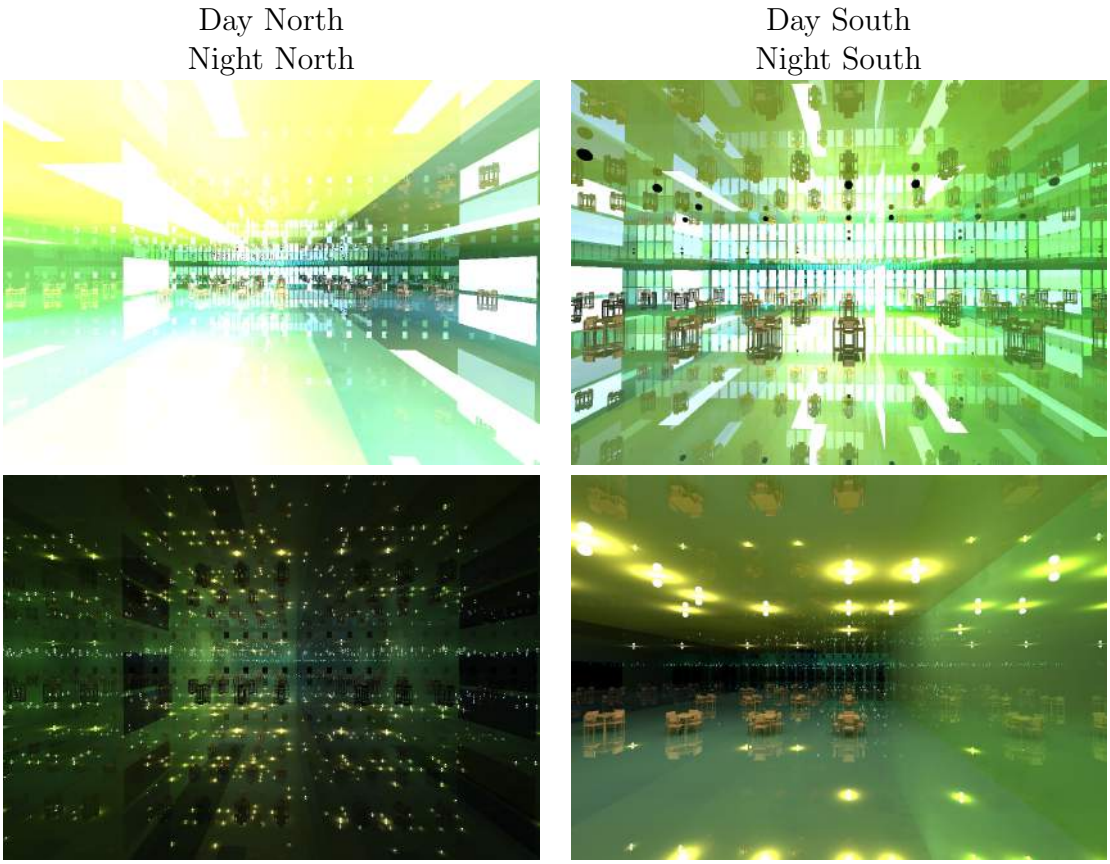

Figure 6.16: Top 5 material generation results cont.

We can see in Figures 6.17, 6.18, and 6.19, evolution of the day middle and dark objectives is fairly flat, and the other 4 objectives are a larger focus of evolution.

	Pre-defined Material	Generated Material
Day Bright	-	-
Day Evenness	✓	-
Day Middle	-	✓
Day Dark	-	✓
Night Bright	-	-
Night Dark	✓	-

Table 6.6: T-test comparison between Day/Night predefined material and Material generation with 90% confidence. A Check mark indicates statistically significant improvement.

A statistical analysis done between this experiment and the pre-defined materials shows a mixed change in performance, as outlined in Table 6.6. The GP generated

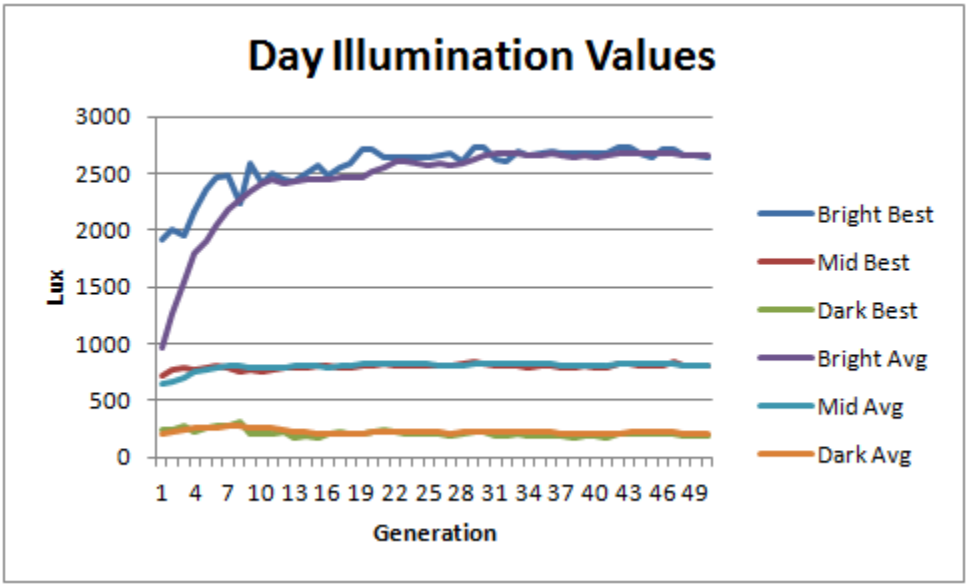

Figure 6.17: The population best and average over 50 generations avg. over 20 runs

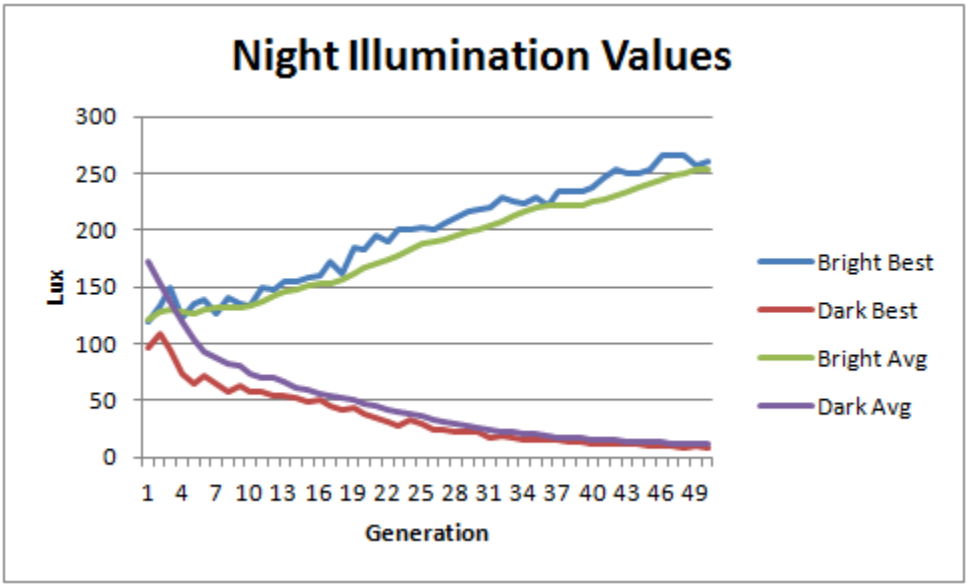

Figure 6.18: The population best and average over 50 generations avg. over 20 runs

materials were able to improve performance during the day in the middle and dark areas of the room. This is most likely because of the lesser use of windows, and instead using the reflected light from the brighter area of the room to meet lighting requirements. The pre-defined materials were shown to perform better for the night time dark area as well as the uniformity. The night dark objective would have

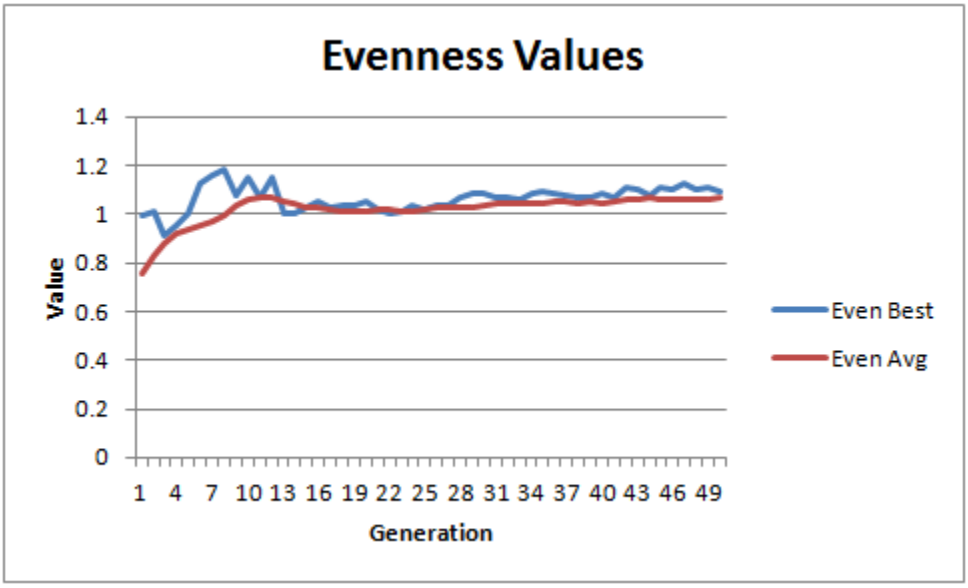

Figure 6.19: The population best and average over 50 generations avg. over 20 runs

been adversely affected by the reflective properties, causing the artificial lights on the opposite end of the room to brighten the dark areas. The uniformity could be attributed to the high glare caused by the materials. The lack of windows would focus the areas which would be exposed to direct natural light. These areas would skew the uniformity measurements.

6.5 Glare Measurement

6.5.1 Setup

The second material experiment adds a new objective to the earlier one: glare. The addition of the glare objective allows the system to have a measurement which controls the material definition. The amount of glare is directly tied to the wall material, and the balance between reflectiveness and discomfort can be controlled.

The measurement used for glare is the optimization of the measurement taken from the Radiance command. Two measurements were taken facing in the north and south directions. The angles of measurement were taken in a 180 degree field of view from the chosen points, with glare calculations taken every 10 degrees, giving 36 value points. This gives a full view of the area. Of these measured values, the lowest value was subject to optimization during evolution. Due to limitations of the software, glare measurements could only be taken during the night phase. Day

time glare measurements would return no results. The cause of this limitation is unknown. The reasoning behind using this limited measurement was that optimizing glare measurements during the night would still affect the material of the room, which is the same used during the day. The same 6 objectives are used as the previous experiment with the one extra glare minimization objective, giving 7 total objectives. The same parameters as Section 6.4 are used as well.

6.5.2 Results

Even with the minimization of the glare levels, the amount of reflectivity still remained high. As seen in Figures 6.20, 6.21, and 6.22, these results were more likely to use side windows for the use of sunlight over the previous experiment. The comparison in Table 6.7 shows that there was an almost even split between the two experiments for performance. This could be attributed to the lack of daylight glare testing in the software. A larger pressure would have been placed on the glare objective if this was available.

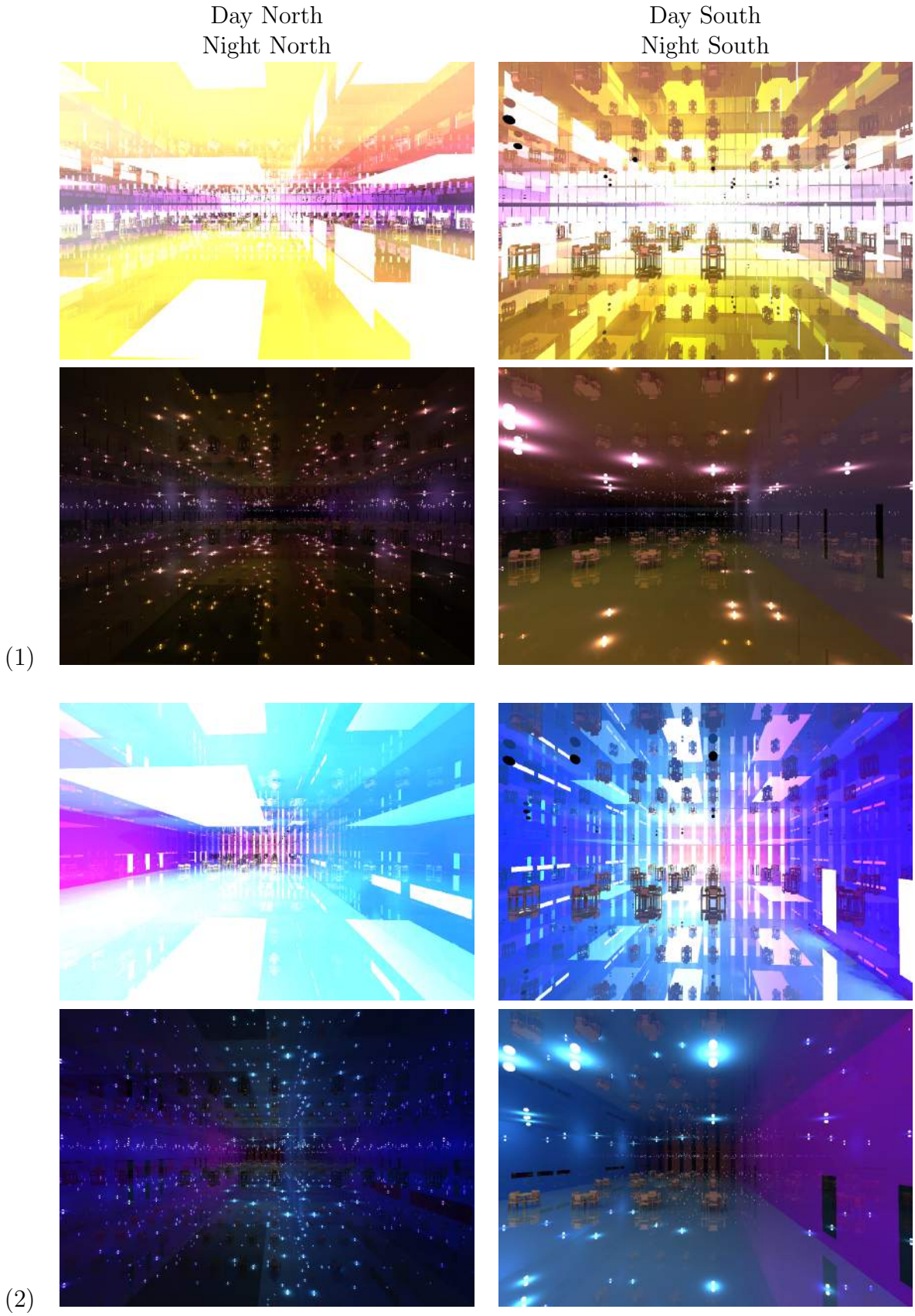

Figure 6.20: Top 5 glare minimization results

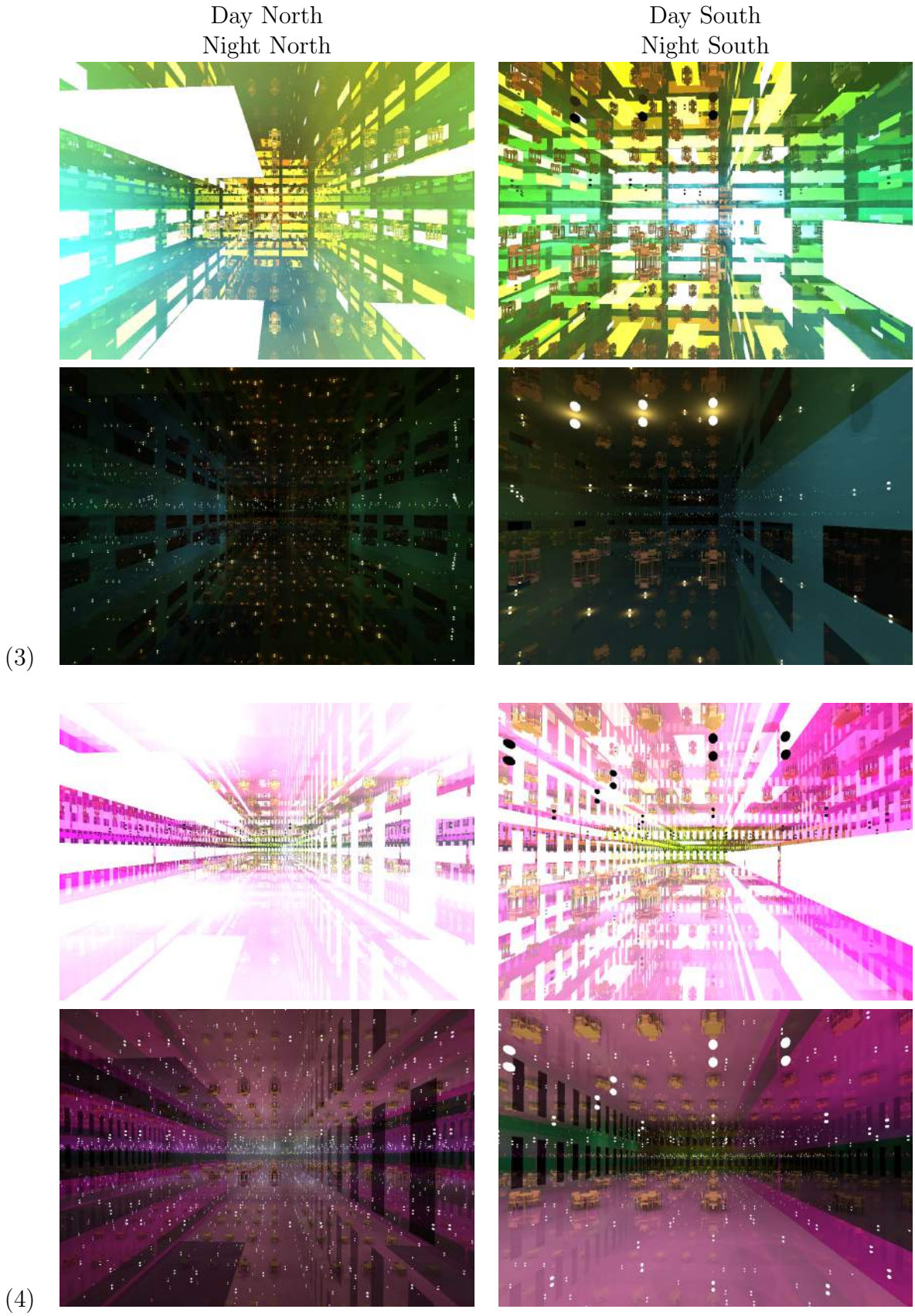

Figure 6.21: Top 5 glare minimization results cont.

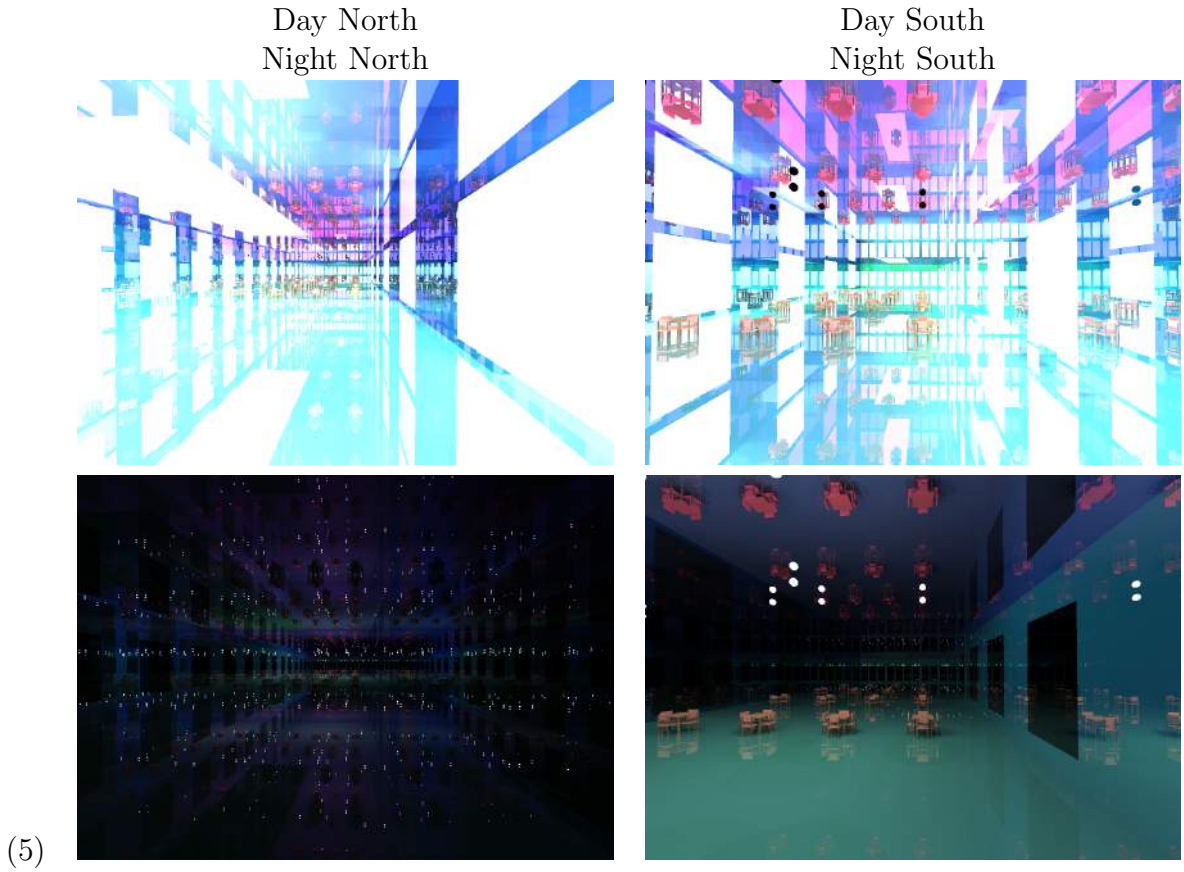

Figure 6.22: Top 5 glare minimization results cont.

	Glare Minimized	Glare Not Minimized
Day Bright	-	-
Day Evenness	-	✓
Day Middle	✓	-
Day Dark	✓	-
Night Bright	-	✓
Night Dark	-	-

Table 6.7: T-test comparison between unconstrained materials and glare minimization with 90% confidence. Check mark indicates statistically significant improvement.

As seen in Figures 6.23, 6.24, and 6.25, the evolution of this experiment concentrates on the bright day, night, and evenness. The medium and dark day objectives are relatively flat in their evolution.

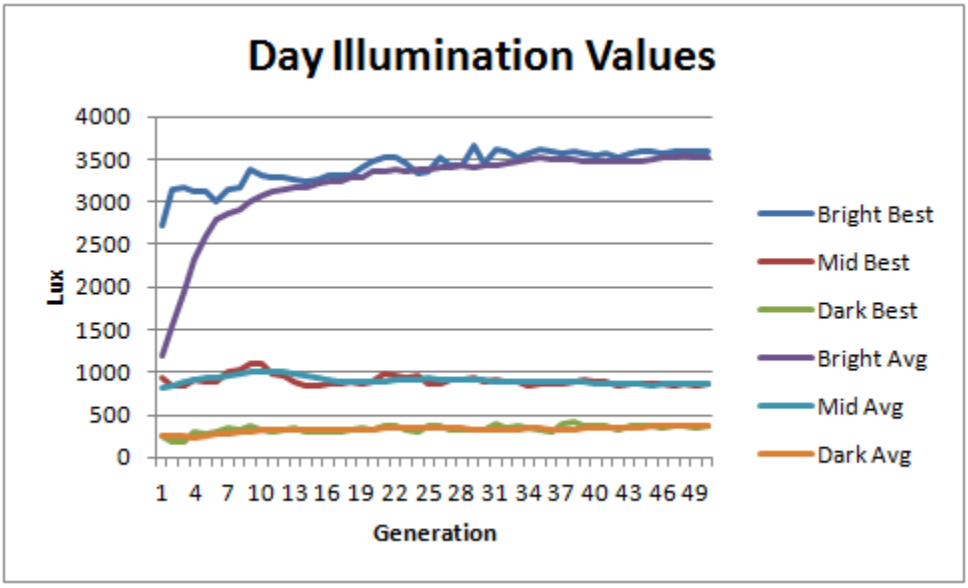

Figure 6.23: The population best and average over 50 generations avg. over 20 runs

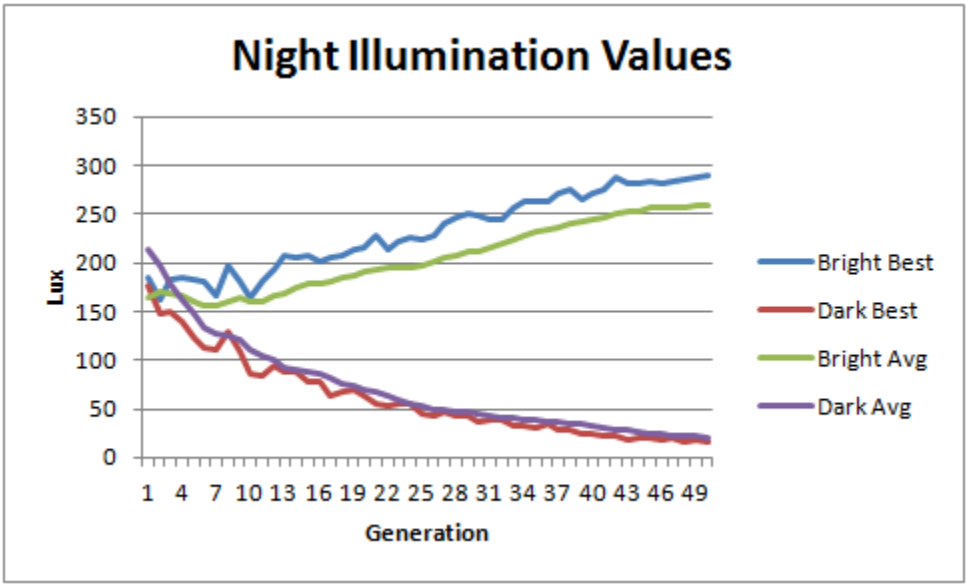

Figure 6.24: The population best and average over 50 generations avg. over 20 runs

6.6 Summary

This chapter was able to cover a variety of different experiments regarding the use of artificial and natural lighting. The use of generated materials were shown to have their strengths as well as weaknesses. Using a greater number of objectives, we were able to show the varying strengths and weaknesses of GP with illumination design.

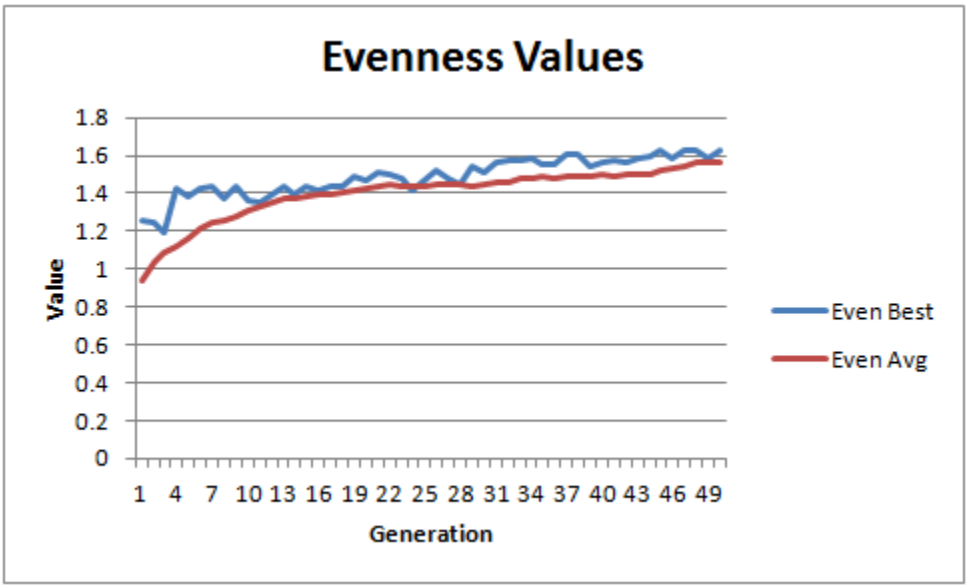

Figure 6.25: The population best and average over 50 generations avg. over 20 runs

Chapter 7

Decorative Illumination

This chapter explores decorative illumination using evolved textures. Two problems are outlined, one using artificial lights and the other with coloured windows. These experiments introduce interesting new applications of evolutionary illumination design.

7.1 Evolution of Coloured Lights

The goal of this section is to generate a grid of decorative coloured lights on a wall through the use of evolved textures.

7.1.1 Setup

Using the lighting analysis tool kit used for the design process of the previous experiments, a new application of evolutionary illumination design is explored. Here, a decorative wall of lights is to be created. A direct analysis of the light being created will be done rather than the standard colour image analysis (used earlier in Chapter 5). To test this method, a matrix of light objects akin to a "texture bitmap" is devised. Each light object is defined with RGB colour channels, allowing the standard creation of colours. GP will evolve colour patterns on the light wall to match a target colour specification.

Radiance uses the high dynamic range (HDR) format for values. Standard colour space measurements are bounded, either between 0 and 255 or 0 and 1. HDR formatting is an unbounded colour format, which can allow colour values to exist in the range from 0 to an upper bound only of the numerical type being used. This posed a problem for the analysis of the textures being created and the mapping which was

desired. A standard mapping could not be directly compared to the HDR values computed from the Radiance program.

To find a solution to this issue, the ratio between colour channels is used as the measurement. This would allow the two separate colour types to be compared without modification to their raw values. The formula used for this function are outlined in Chapter 4. The process behind this method was to preserve the raw colour hue. The difference between a colour of one value and that of another with the same channel ratios would simply be the brightness of light which is generated, but not the colour itself.

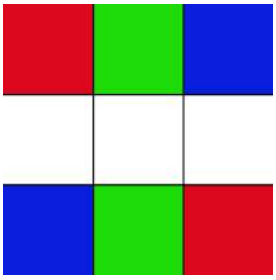

Figure 7.1: Target colour map

Type	Function
LW	Light Wall
I	ERC Int
TF	X pos, Y pos
TF	+, -, *, ÷ sin, cos, log, neg

Table 7.1: GP Language for texture generation

The functions in Table 7.1 outlines the GP language used for the coloured light wall. Given a blank, windowless wall, the GP system was given control over the colours of the lights as well as the resolution of the grid to be used, with a minimum of 3×3, and a maximum of 10×36. This gave the lights a minimum distance apart of 0.5 metre. Colour channels were generated through three mathematical expression trees, with each channel generating a separate expression tree. Measurements of the lights were taken at 9 equal distant grid points, placed directly in front of selected light objects. A target colour map was used as a measurement of fitness, as shown in Figure 7.1. Having the GP system focus on the evolution of specific points rather than the texture as a whole will allow a greater degree of freedom in the creativity of the answers.

The function used for creating the lights is "Coloured_Wall(I,I,TF,TF,TF)". This function uses two integers to determine the width and height dimensions of the grid. The dimensions are found through the modulo function on the values, giving a value between 1 and 36 for the width and 1 and 9 for the height. Each tree float argument represents a colour channel, one for each of red, green, and blue. Terminals for these trees are taken from the X and Y position of the current light.

7.1.2 Results

As seen in Figures 7.2, 7.3, 7.4, and 7.5, which are the best solutions from each of the 20 runs, the matrix of lights are able to create the rudimentary semblance of a texture. The restricted resolution of the grid had a significant effect on the possible outcomes. The relatively low resolutions did not allow for enough variance in the evolved texture formula. This caused the solutions to make sacrifices to fulfil the requirements of fitness. It is possible that as the resolution is increased that the solutions will fit the target mapping better, but as the resolution increases, the interference between light objects becomes greater. Fitting many light producing objects into a limited space can cause the image to become muddied, washing out the intended image. The interaction and interference between light objects was amplified by the low resolution of the wall. A solution would be to lower the brightness of each object to where their interaction is minimized, but this would eliminate one of the focuses of the experiment. Having many small dim objects would be very similar in style to a regular bitmap texture generation. It is the brightness of the objects simulating light bulbs which is of interest. An example of the GP tree generated from this experiment can be found in Appendix A.

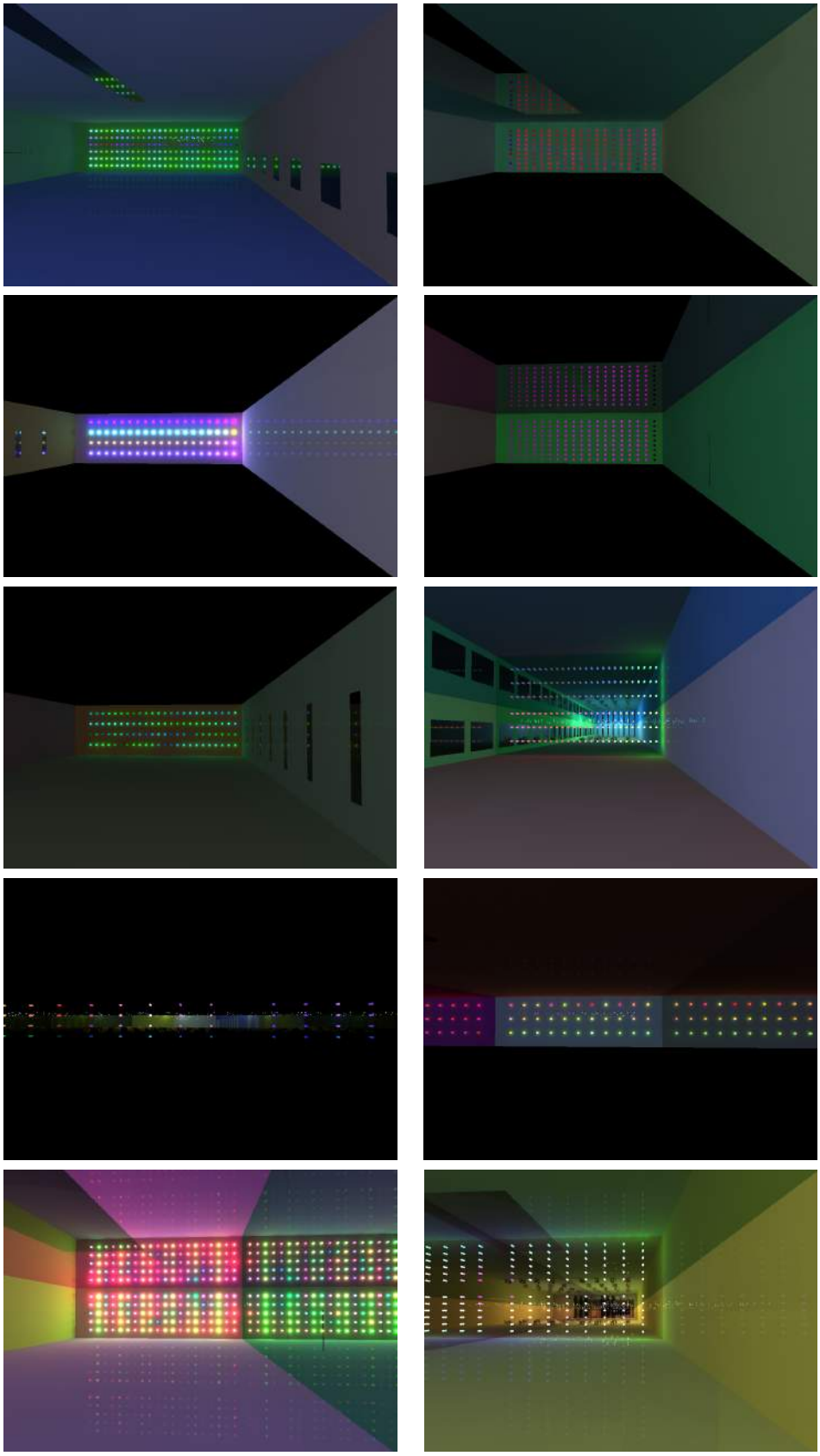

Figure 7.2: Light wall results

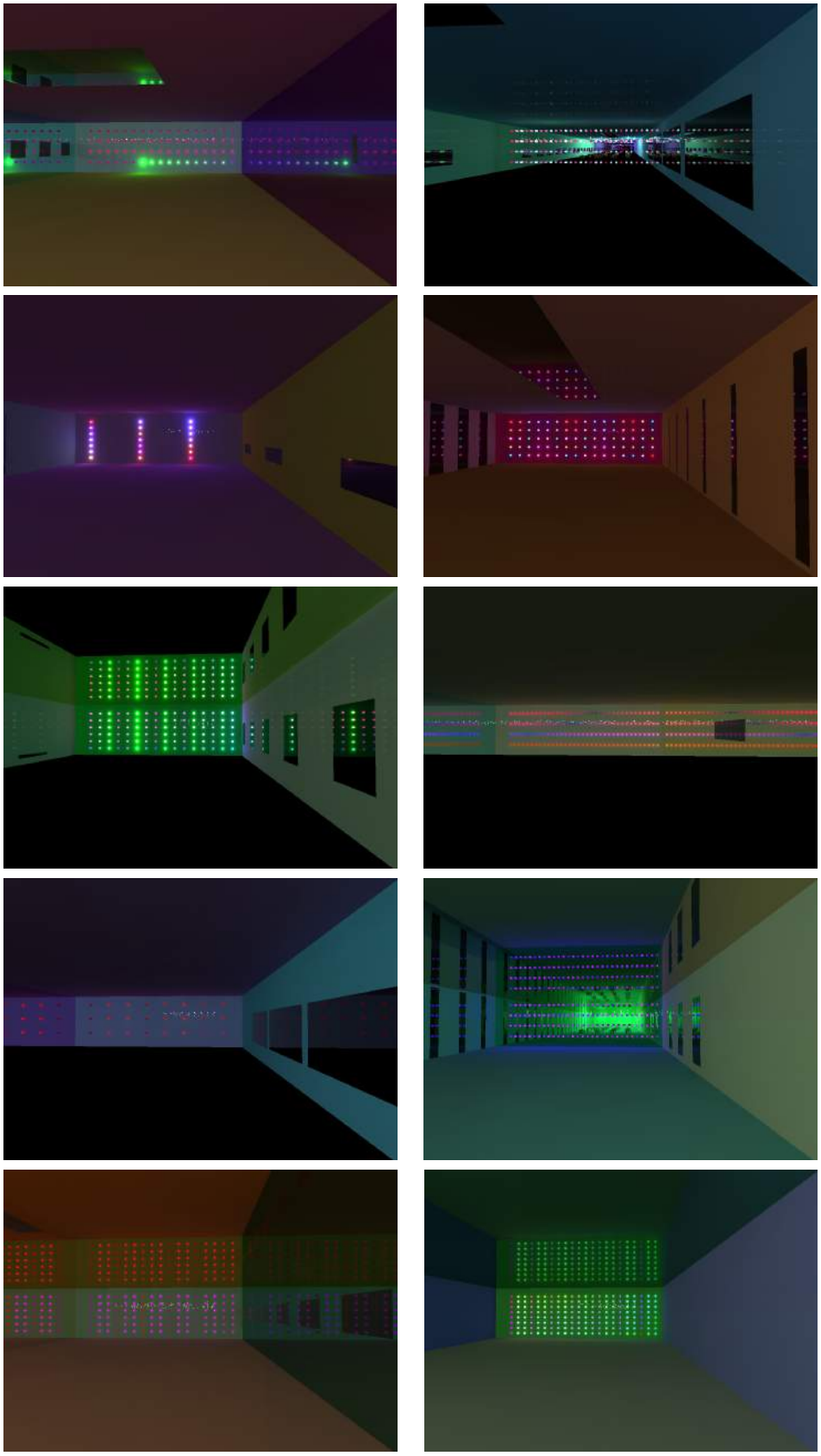

Figure 7.3: Light wall results cont.

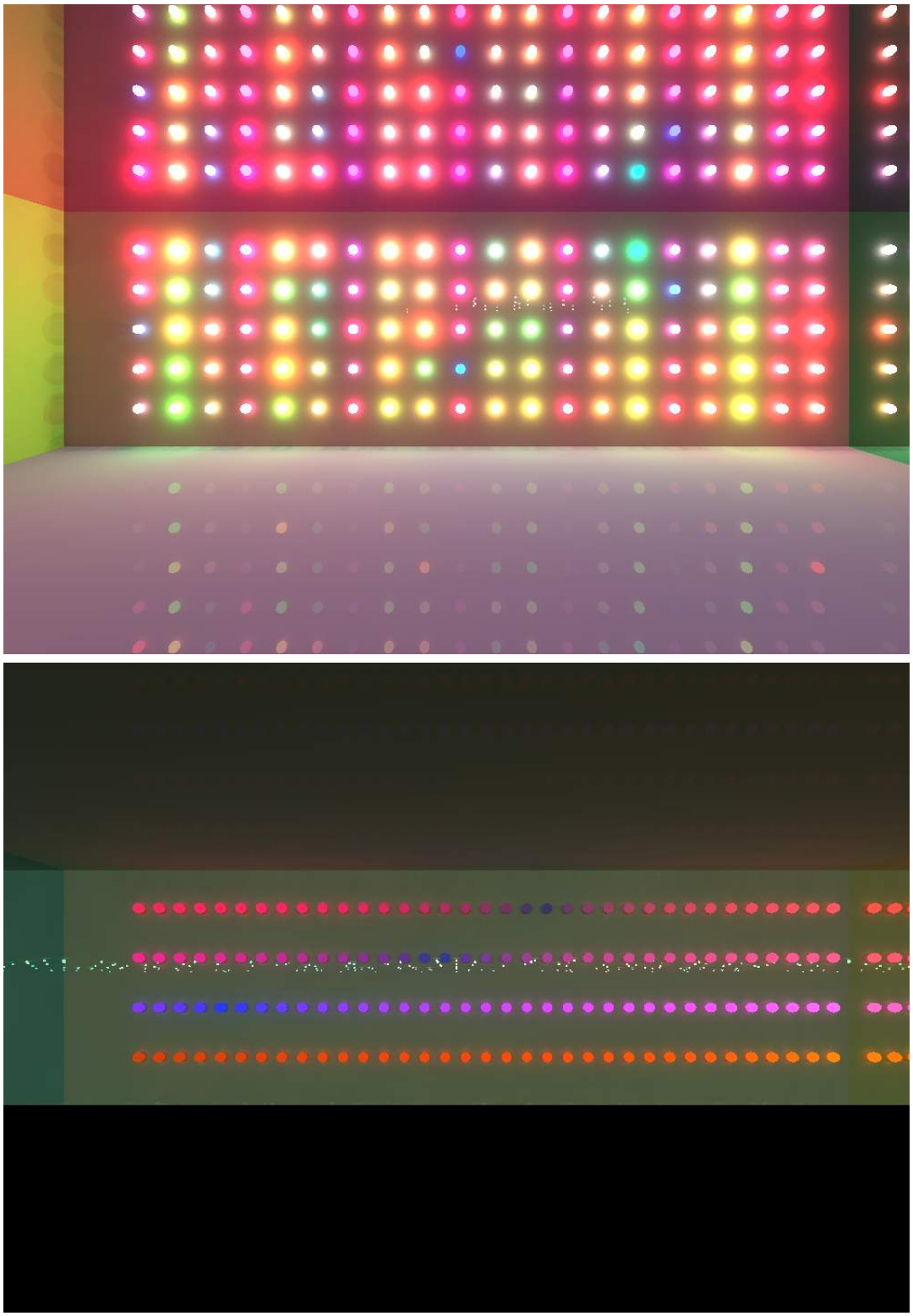

Figure 7.4: Examples of light wall results.

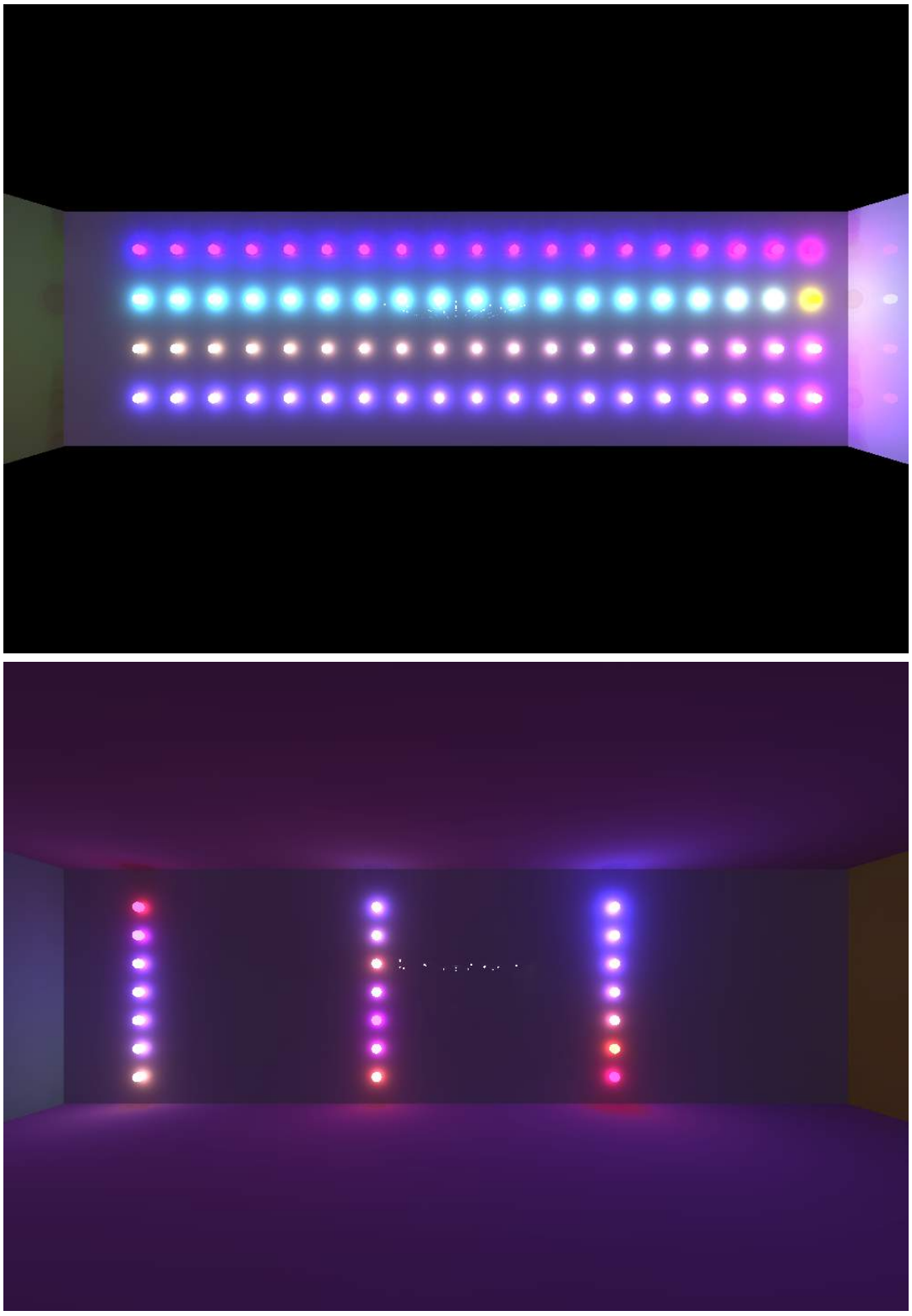

Figure 7.5: Examples of light wall results.

7.2 Coloured Glass and GP texture creation

Next, GP is used to evolve a decorative stained glass window. The simulation of stained glass is an interesting and practical goal. Unlike the artificial lights, the resolution of a stained glass window could be expanded to much higher numbers. This was because the texture being generated would be created using glass material objects. This material is defined by the transmittance of each colour channel through the object, and does not create the light itself. This helps to reduce the cluttering and interference of multiple close lights as seen before.

7.2.1 Setup

The GP function trees for this experiment uses a coordinate system centred on the origin, where $-1 \leq i \leq 1$ for x and y, to put the origin in the middle of the window, resulting in a center based texture. The function used in creating the lights is "Stained_Glass(I,I,TF,TF,TF)". This function uses two integers to determine the width and height dimensions of the grid. The dimensions are a random number between 50 and 100. Each tree float argument represents a colour channel, one for each of red, green, and blue, and generates a colour for the given (x, y) coordinates.

An environment in a large building was created as a stage for the window creation. The southern side of the building was set for the placement of the window. The setup was to have two images to be viewed: the original stained glass window on the wall, as well as the image projected into the room by the sun. The projection on the floor would be the area used for colour target matching. This is done to make indirect colour analysis, which is more challenging then the direct measurements used for the wall of lights. The target points were set above the floor looking downward from the south at a 45° angle. The light captured at these points would be the reflection of the image to the viewer, reflecting how the image would be viewed by human eyes with their head towards the window. Figure 7.6 demonstrates how the collection is done.

To create a suitable sized image projection, the solar noon of the autumnal equinox of 2014 at the location of 43.119° north 79.245° west was used, the location of Brock University's Computer Science Department. Solar noon is the time of day where the sun crosses the celestial meridian, placing the sun directly south. The resolution of the window is allowed a resolution of between 50×50 and 100×100 panels, giving a total possible count of 10,000 pieces of glass. As before, a sum of errors was used for fitness evaluation. The testing points as well as their target colours are shown in Figure 7.7. The target map is the same target map used in the previous experiment,

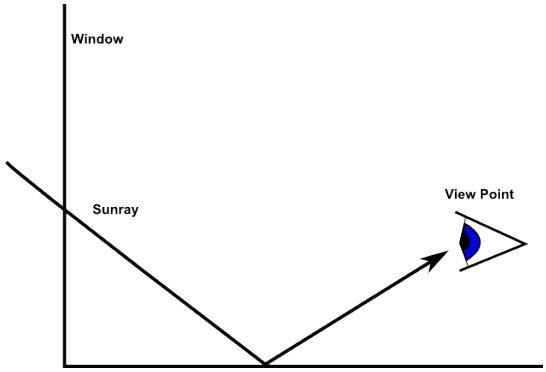

Figure 7.6: Light collection example

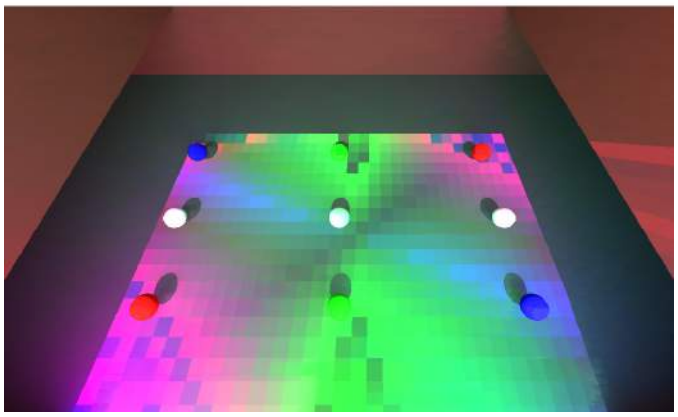

Figure 7.7: Target colours placed in target points

shown in figure 7.1.

7.2.2 Results

The higher resolution was able to give much better results than the previous experiments. Texture generation equations were able to work to a more full potential, creating interesting and intricate patterns. The measurement style of indirect light allows for a different view of texture generation. Figure 7.8 shows the differentiation between the glass and the projection it creates. While similar, they have differences that show how the sunlight causes the interaction of the diffuse lighting. In Figures 7.9, 7.10, 7.11 and 7.12, which are the best solutions from each of the 20 runs, we can see certain patterns emerge in the results. Many of the results create striking lines on diagonals of single colours. This helps to fulfil certain colour requirements. When

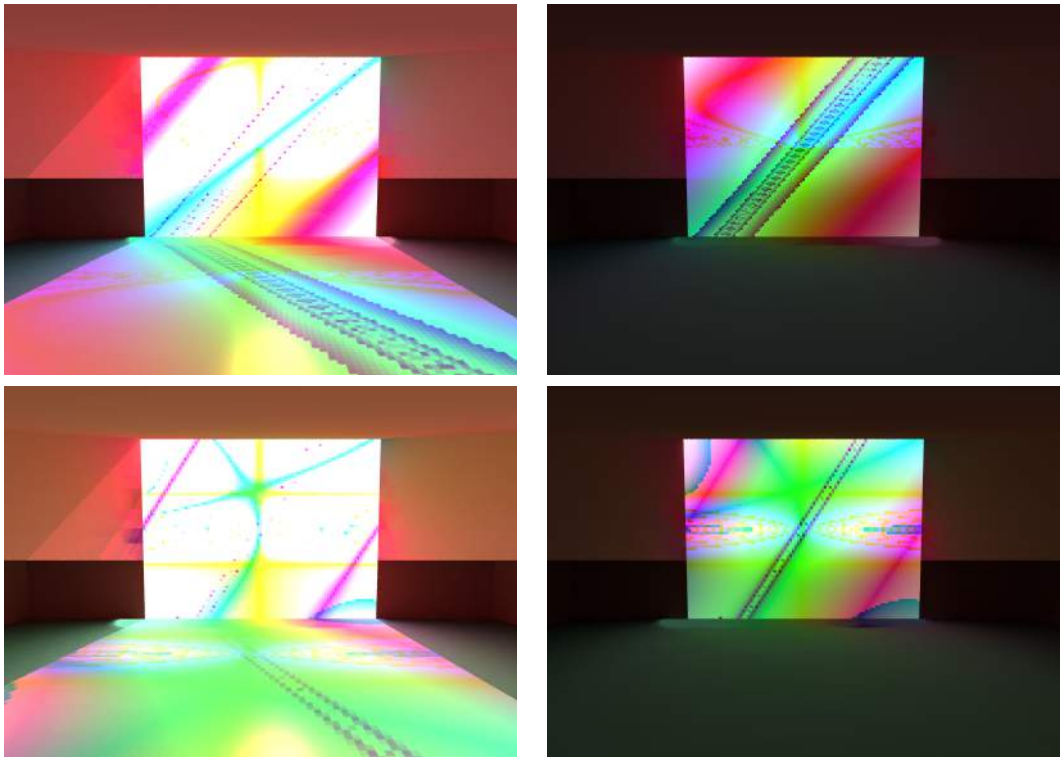

Figure 7.8: Mid day and late day

this happens, the alternate corners will take to the other target corner colour. The middle green is commonly a straight vertical line in the middle of the window. To meet the white light requirements in the middle horizontal row, the colours will begin to blend before diverging again for the colours. An example of the GP tree generated from this experiment can be found in Appendix A.

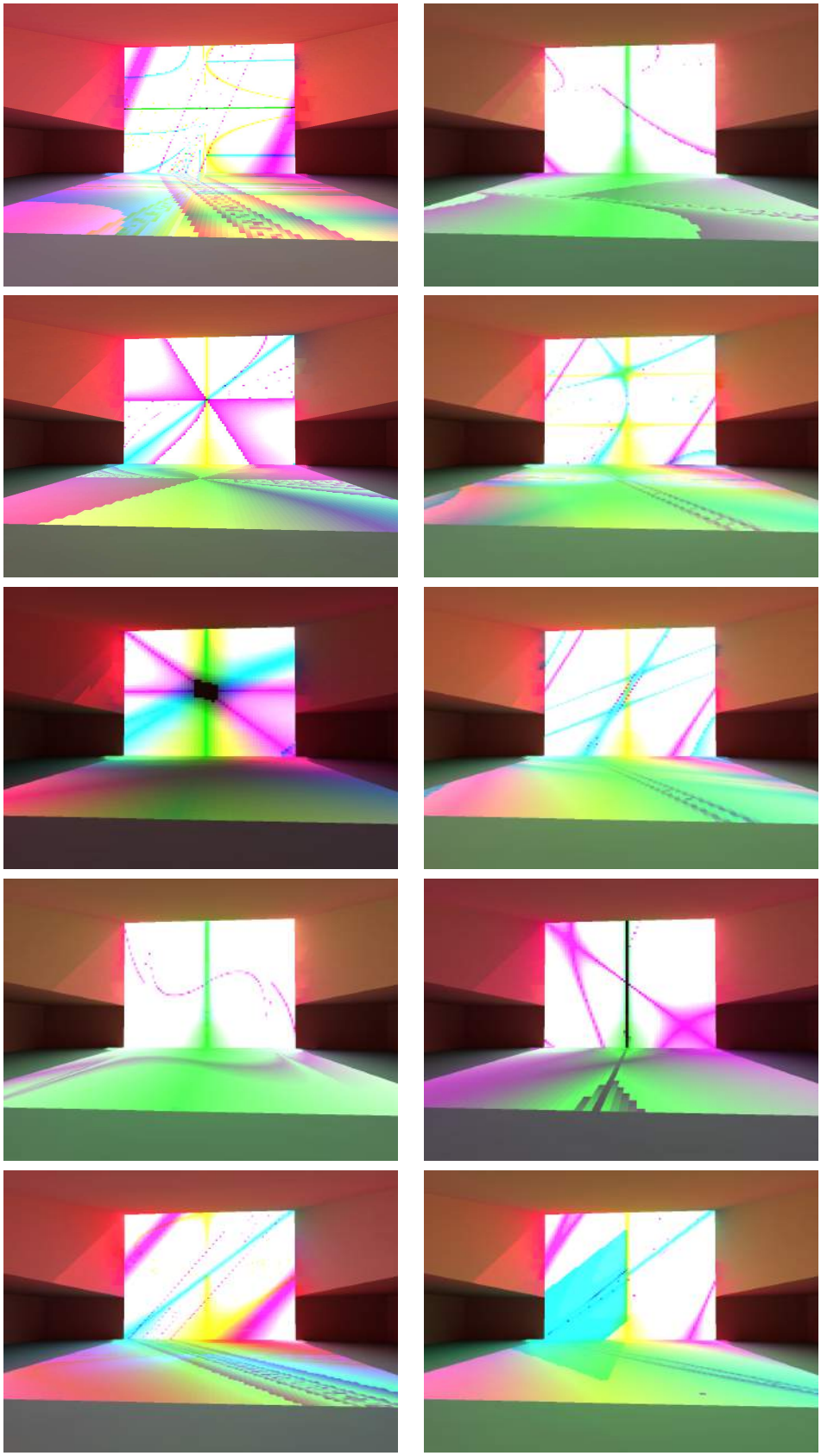

Figure 7.9: Stained glass results

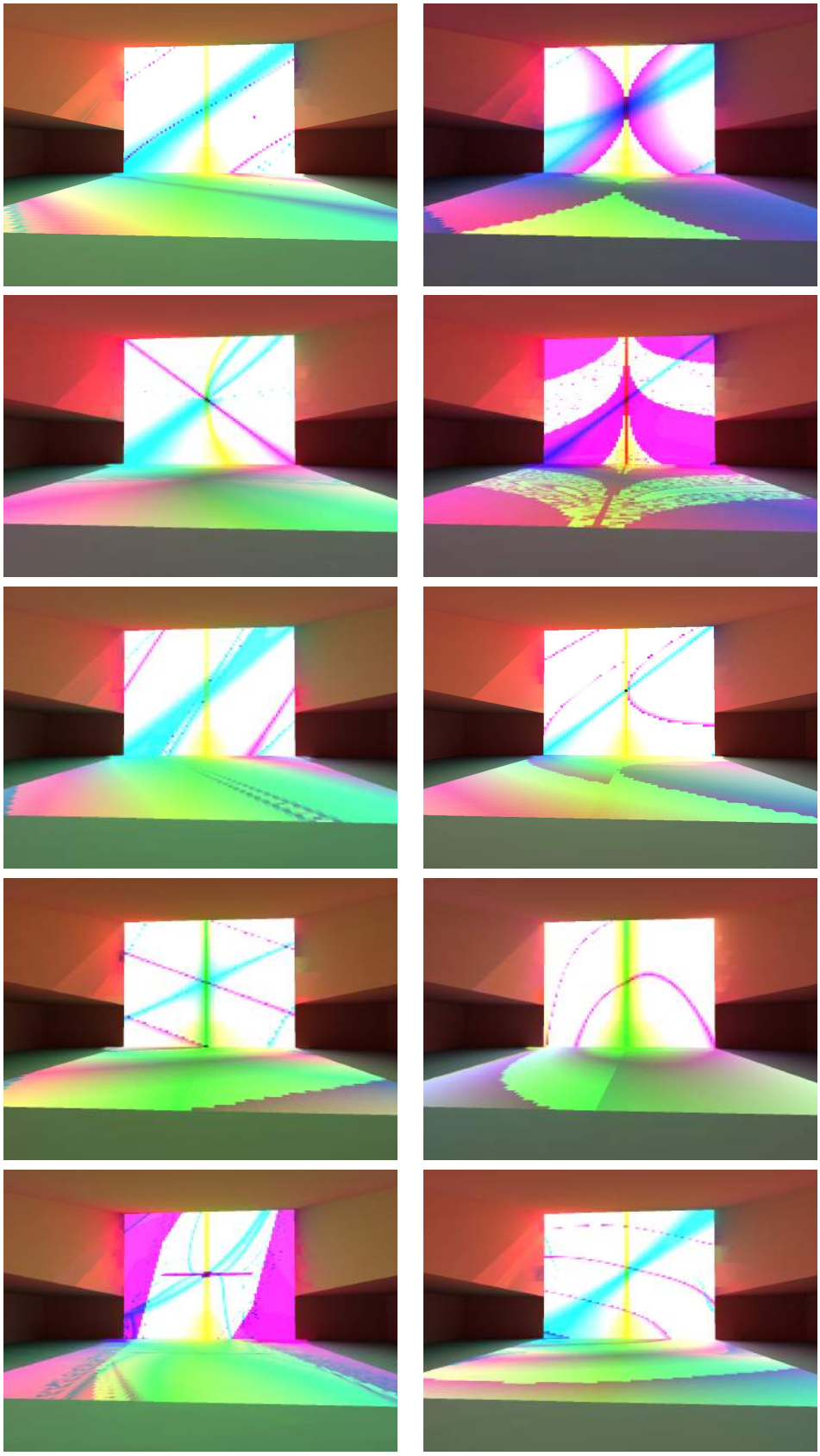

Figure 7.10: Stained glass results cont.

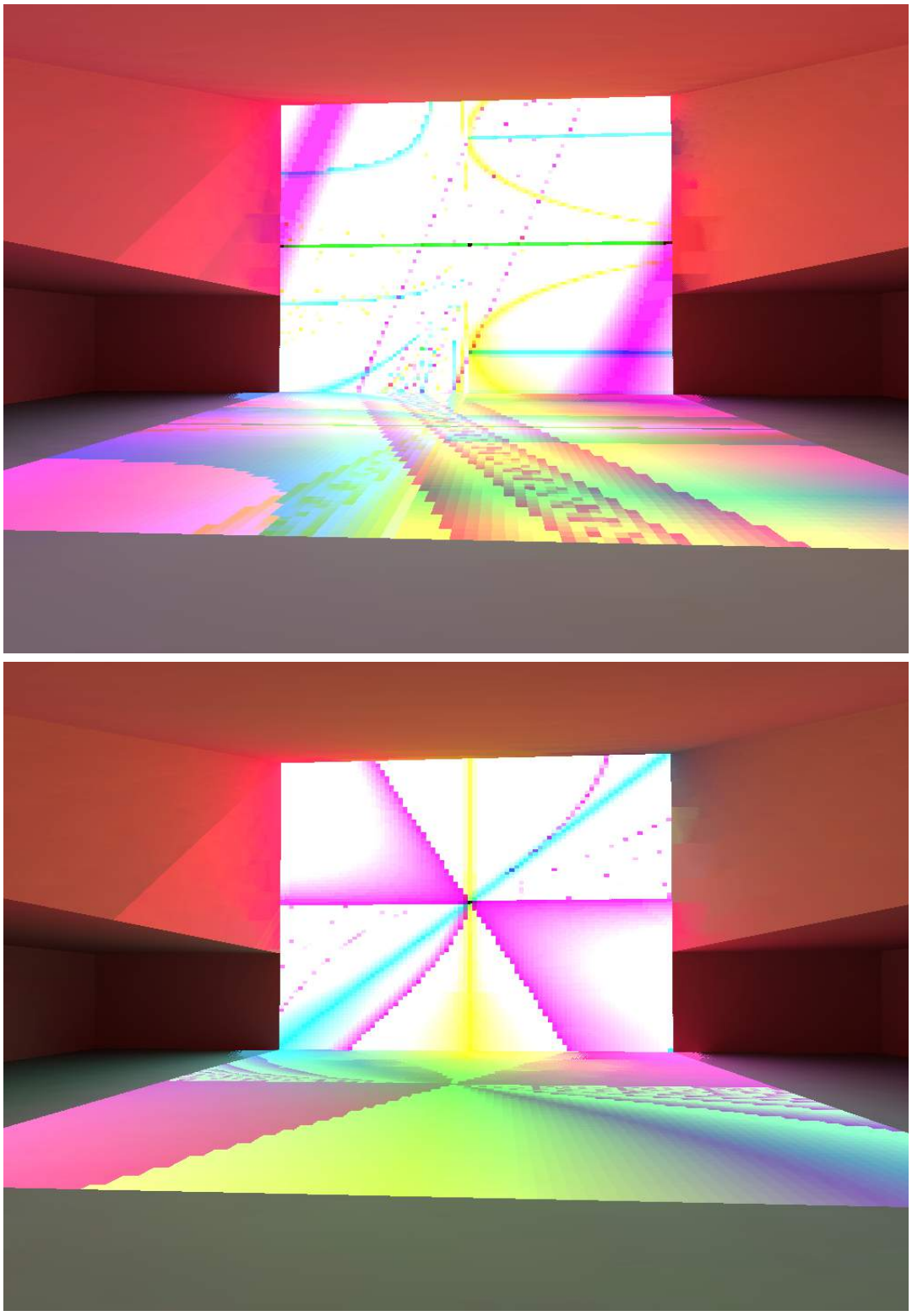

Figure 7.11: Examples of stained glass results

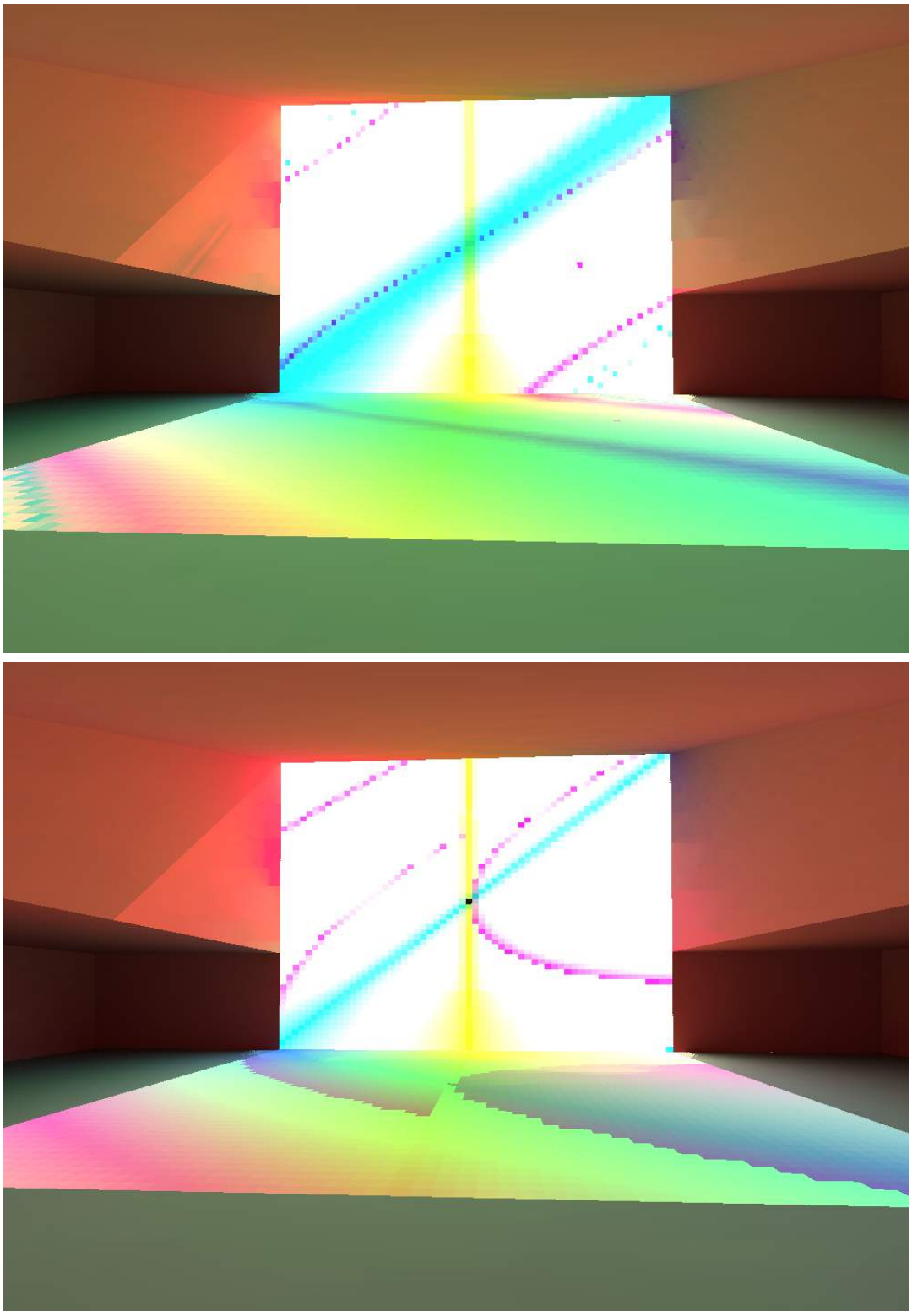

Figure 7.12: Examples of stained glass results cont.

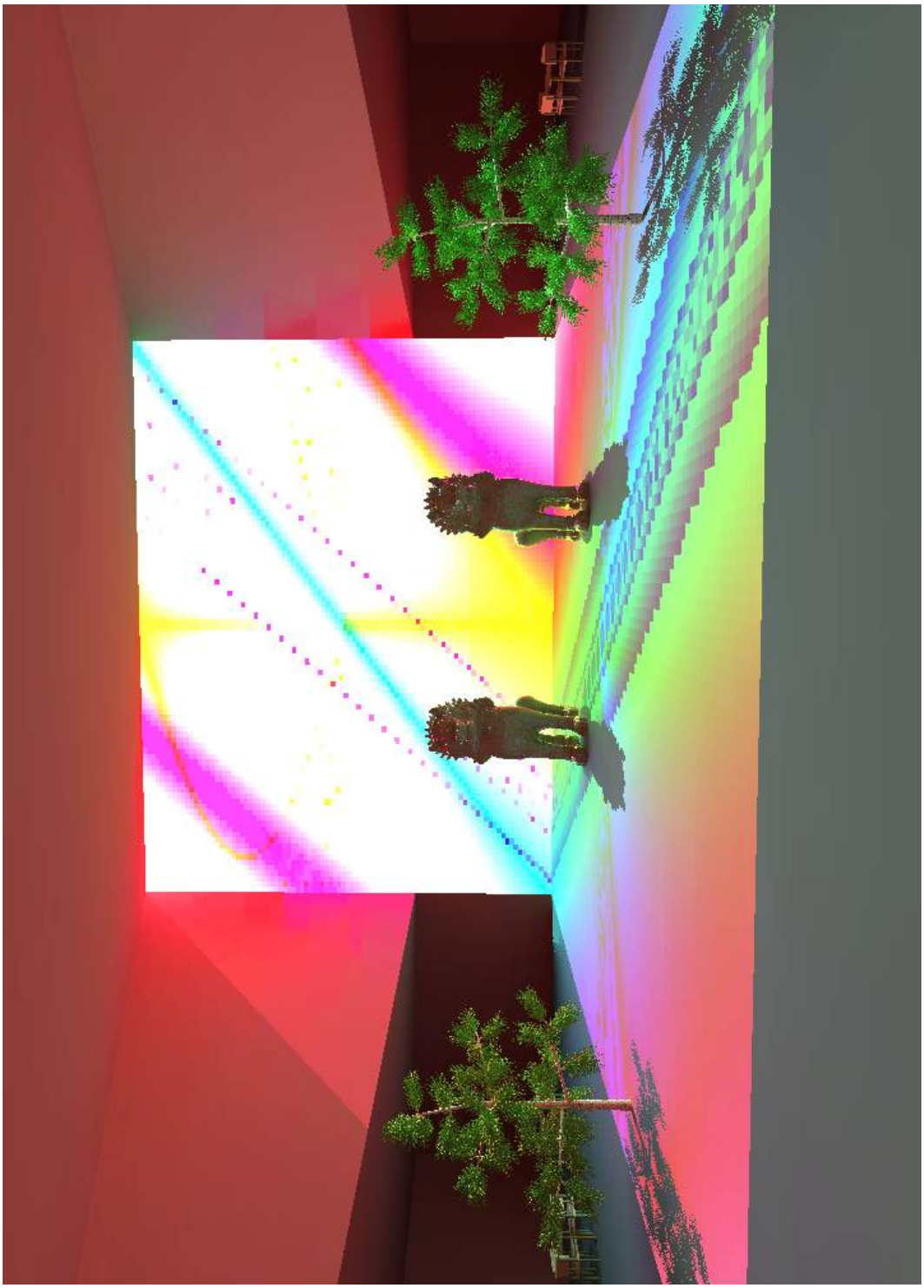

Figure 7.13: High resolution rendering of solution

7.3 Summary

In this chapter we introduced new interesting problems for both illumination design and texture generation. We showed that the use of artificial lights can be challenging. The indirect light analysis used in the stained glass resulted in very interesting creations.

Chapter 8

Comparisons

This chapter will discuss the results of the previous experiments, comparing them to research done previously in the field of inverse illumination design.

We used an unsupervised genetic programming approach for finding target matching solutions on surfaces, in comparison to research done by Tena[40], which used interactive evolution.

Our research used skylights to meet lighting requirements, in similar fashion to Fernandez *et al.*[13]. Both research used many objectives to optimize a given scene. Our approach was evolutionary, while Fernandez *et al.* used both non-evolutionary as well as evolutionary techniques.

Costa *et al.*[9] also used Radiance for light analysis. They looked at geometry, material properties, and had a design goal with the use of artificial lights. We also looked at these factors, but Costa *et al.* used simulated annealing as their search technique rather than an evolutionary approach.

Shea *et al.*[34] used ACO for building envelope design. They used skylight type designs of roofs to maximize for target daylight use. While our work used a skylight as well, we used skylights in conjuncture with windows on side walls, the focus of Shea *et al.* was on ceilings without a focus on windows. Our work focused on using windows for allowing light into the room. The technique for placing windows was similar, with both works allowing the systems to choose between the placement of glass and opaque panels.

Mourshed *et al.*[36][35] looked at both artificial and natural lighting, though in separate studies. Our work combined the two types of lighting. Their natural lighting study used an exhaustive search, while their artificial lighting study was more similar to our research in using an evolutionary approach. In both studies, they used a grid of test points to gather light data, and optimized for the evenness of the lighting

solution. We used both of these techniques successfully as well.

Oraei[16] used GP to design buildings for energy efficiency rather than illumination requirement. The work used material generation window creation, in a more complex degree than ours. He found solutions which used passive solar energy to meet efficiency requirements, in the same way our work used natural light to fulfil lighting requirements.

Castro *et al.*[7] compared a number of different heuristic and evolutionary approaches rather than focusing on a single technique for multiple problems. They evolved placements of light source objects, similar to our artificial lighting objects, to meet their lighting requirements as well as efficiency targets.

Caldas[6] also looked at illumination as a many objective problem, though her research used a GA rather than GP. She evolved all aspects of the building, including windows and ceilings, in similarity to our work. Her work did not include the use of artificial lighting like ours, and focused on the creation of a whole building, rather than a single room. We used a normalized sum of ranks for fitness evaluation while Caldas used a pareto based GA.

Chapter 9

Conclusion

This chapter gives a brief summary of the research that was done as well as the results we obtained. It outlines potential future work which could be used to expand the research, and areas of interest for further research.

9.1 Results

This research investigated the use of genetic programming on inverse illumination problems. It used a variety of fitness measurement techniques to evaluate a number of illumination factors of various levels of difficulty. GP was shown to be an effective approach towards solving the illumination problems. As the number of objectives increased, GP was able to find satisfactory solutions for the growing complexity of the problems. As many as 7 objectives were used at once. It was shown that the combination of artificial and natural lighting sources could produce a variety of solutions to differing problem sets. The inclusion of evolved materials and glare measurements gave creative insights into possible variations of room design.

The combination of GP texture generation, a highly studied field, and illumination analysis with the wall of decorative lights and stained glass proved that it could produce new and interesting results for decorative illumination design problems. We were able to create a wide variety of solutions for textures using an indirect colour sampling technique, as well as showing an effective technique for comparison between HDR and the standard RGB colour space. The introduction of these two new interesting problems opens a wide range of possibilities for research.

A major issue found during the work was the computational cost of realistic light calculations. The Radiance system was not designed for such high volume computations, and as a result the experiment times grew greatly as the complexity increased.

The longest of the experiments, those which used glare measurements, took upwards of 300 hours or more of computation time per run. The software was also limited in these glare measurements in relation to natural light. This limitation caused some experiments to be less robust than they could have been. This could be due to how Radiance has been used by designers since its creation. It has been adapted into many other simulation software systems, and is not often directly used and implemented as was done here. It is possible that these issues of computational cost could be fixed by adapting the simulation algorithms directly from Radiance into the GP system, but that would require a great amount of work.

9.2 Future Work

There are many potential future extensions and applications of this research. New design objectives and factors can be included. A logical next step would be the inclusion of an energy efficiency objective, such as passive solar energy. The expansion of the environment into a more complex 3D structure would also be a good natural enhancement. While these experiments are all run on large empty halls, most rooms have more complex structure to them. Multiple rooms, and even multiple levels of a building, would be ways to make the problem more realistic. It is very rare for a designer to design a single large room, and instead looks at the relation between multiple rooms for how they work together. To maintain realism, commercially available objects used in evolution can be added, such as lights, materials, paints, glass, etc. More advanced window creation techniques could be developed. Rather than uniform rectangular windows, more shapes and advanced generation techniques could be used to create more complex window designs.

Aesthetics is another area of focus which could be considered. While the optimal use of light is a desirable factor in interior design, it does not take into account the advanced aesthetic considerations of the use of that room that a human interior designer would consider. Factors such as colour relations, mood, or sense of space are much harder to translate into a computational form. These human competitive concepts in interior design are large hurdles which should eventually be addressed.

The use of evolved textures in illumination has many potential extensions for future research. The field of GP texture generation has been greatly explored, and results can be applied to illumination problems. Noise filters, entropy, and other complex functions can be added to the texture language for more interesting results. Models of aesthetics could also be taken into consideration for texture evolution[12].

More realistic and traditional stained glass generation could also be attempted. Real stained glass is created with bold outlines and non-uniform shapes to create images rather than textures.

Bibliography

[1] Daniel Ashlock, Balu Karthikeyan, and Kenneth M Bryden. Non-photorealistic rendering of images as evolutionary stained glass. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2087–2094. IEEE, 2006.

[2] Peter Bentley and David Corne. *Creative evolutionary systems*. Morgan Kaufmann, 2002.

[3] Peter J Bentley and Jonathan P Wakefield. *Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms*. Springer, 1998.

[4] Steve Bergen and Brian J Ross. Aesthetic 3d model evolution. *Genetic Programming and Evolvable Machines*, 14(3):339–367, 2013.

[5] Stephen Brooks. Image-based stained glass. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1547–1558, 2006.

[6] Luisa Caldas. Generation of energy-efficient architecture solutions applying gene_arch: An evolution-based generative design system. *Advanced Engineering Informatics*, 22(1):59–70, 2008.

[7] Francesc Castro, Esteve del Acebo, and Mateu Sbert. Energy-saving light positioning using heuristic search. *Engineering Applications of Artificial Intelligence*, 25(3):566–582, 2012.

[8] David W Corne and Joshua D Knowles. Techniques for highly multiobjective optimisation: some nondominated points are better than others. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 773–780. ACM, 2007.

[9] António Cardoso Costa, António Augusto Sousa, and Fernando Nunes Ferreira. Lighting design: A goal based approach using optimisation. In *Proceedings of*

the 10th Eurographics conference on Rendering, pages 317–328. Eurographics Association, 1999.

[10] Barbara Cutler, Yu Sheng, Steve Martin, Daniel Glaser, and Marilyne Andersen. Interactive selection of optimal fenestration materials for schematic architectural daylighting design. *Automation in Construction*, 17(7):809–823, 2008.

[11] Richard Dawkins. *The Blind Watchmaker: Why The Evidence Of Evolution Reveals A Universe Without Design Author: Richard Dawkins, Publisher*. WW Norton & Company, 1996.

[12] E. den Heijer and A.E. Eiben. Comparing aesthetic measures for evolutionary art. In *Proc. EvoMusArt*, volume 2, pages 311–320. Springer, 2010. LNCS 6025.

[13] Eduardo Fernández and Gonzalo Besuievsky. Inverse lighting design for interior buildings integrating natural and artificial sources. *Computers & Graphics*, 36:1096–1108, 2012.

[14] Robert Flack. Robgp - robust object based genetic programming system. http://sourceforge.net/projects/robgp/, September 2009.

[15] Ladan Ghobad, Wayne Place, and Jianxin Hu. The impact of systems integration on the daylighting performance of skylights in offices. In *WREF Conference Proceedings*, Denver, CO., 2012. American Solar Energy Society.

[16] Mohammad MO Gholami and Brian J Ross. Passive solar building design using genetic programming. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 1111–1118. ACM, 2014.

[17] David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989.

[18] Jeanine Graf and Wolfgang Banzhaf. Interactive evolution of images. In *in Proc. Intl. Conf. on Evolutionary Programming*, pages 53–65, San Diego, CA, 1995.

[19] Anat Grynberg. Validation of radiance. *Lawrence Berkeley Laboratory, Applied Sciences Division, Lighting Systems Research Group, Report# LBID*, 1575, 1989.

[20] Sylvester K Guth. Computing visual comfort ratings for a specific interior lighting installation. *Illuminating Engineering*, 61(10):634, 1966.

[21] Richard Kittler and Peter Valko. Radiance distribution on densely overcast skies: comparison with cie luminance standard. *Solar Energy*, 51(5):349–355, 1993.

[22] Hank Koning and Julie Eizenberg. The language of the prairie: Frank lloyd wright's prairie houses. *Environment and Planning B*, 8(3):295–323, 1981.

[23] John R Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science, 1990.

[24] Greg Ward Larson. http://radsite.lbl.gov/radiance/gallery/daylite.html, 1994. Date Accessed: 2013-09-26.

[25] Greg Ward Larson, Daniel E Fuller, and Andrew McNeil. Radiance. http://www.radiance-online.org/.

[26] Greg Ward Larson, Rob Shakespeare, Charles Ehrlich, John Mardaljevic, Erich Phillips, and Peter Apian-Bennewitz. *Rendering with radiance: the art and science of lighting visualization*. Morgan Kaufmann, San Francisco, CA, 1998.

[27] Philippe Marin, Jean-Claude Bignon, Hervé Lequay, et al. Generative exploration of architectural envelope responding to solar passive qualities. In *International Conferences on Design and Decision Support Systems in Architecture and Urban Planning*, 2008.

[28] Giovanni Moretti, Paul Lyons, and Stephen Marsland. Computational production of colour harmony. part 1: A prototype colour harmonization tool. *Color Research & Application*, 38(3):203–217, 2013.

[29] Azza Nabil and John Mardaljevic. Useful daylight illuminance: a new paradigm for assessing daylight in buildings. *Lighting Research and Technology*, 37(1):41–57, 2005.

[30] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning layouts for single-pagegraphic designs. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1200–1213, 2014.

[31] Gustavo Patow and Xavier Pueyo. A survey of inverse rendering problems. In *Computer graphics forum*, volume 22, pages 663–687. Wiley Online Library, 2003.

[32] Brian J Ross, William Ralph, and Hai Zong. Evolutionary image synthesis using a model of aesthetics. In *Proceedings CEC 2006*, pages 1087–1094, Vancouver, BC, 2006. IEEE.

[33] Sage Russell. *The architecture of light: architectural lighting design concepts and techniques*. Conceptnine, 2008.

[34] Kristina Shea, Andrew Sedgwick, and Giulio Antonuntto. Multicriteria optimization of paneled building envelopes using ant colony optimization. In *Intelligent Computing in Engineering and Architecture*, pages 627–636. Springer, 2006.

[35] S Shikder, Monjur Mourshed, and A Price. Luminaire position optimisation using radiance based simulation: a test case of a senior living room. In *Proceedings of the International Conference on Computing in Civil and Building Engineering, Nottingham*. The University of Nottingham, 2010.

[36] S Shikder, Monjur Mourshed, and A Price. Optimisation of a daylight-window: hospital patient room as a test case. In *Proc. of the International Conference on Computing in Civil and Building Engineering, Nottingham, UK*, pages 387–390, 2010.

[37] Karl Sims. Artificial evolution for computer graphics. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, volume 25, pages 319–328, New York, NY, USA, 1991. ACM.

[38] Koen Steemers, Nick Baker, David Crowther, Jo Dubiel, and Marialena Nikolopoulou. Radiation absorption and urban texture. *Building research & information*, 26(2):103–112, 1998.

[39] H. Takagi. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[40] Joaquın Elorza Tena, Isaac Rudomin, Av Eugenio, Garza Sada, and Cerro Gordo. An interactive system for solving inverse illumination problems using genetic algorithms. *Proceedings of computación visual*, 1997.

[41] Stephen Todd and William Latham. *Evolutionary art and computers*. Academic Press, Inc., 1994.

[42] Michela Turrin, Peter von Buelow, Rudi Stouffs, and Axel Kilian. Performance-oriented design of large passive solar roofs. *A method for the integration of parametric modeling and genetic algorithms. in: Proceedings of eCAADe*, pages 321–330, 2010.

[43] Graeme Watt. Synthesis design and visualisation limited. http://radsite.lbl.gov/radiance/gallery/daylite.html. Date Accessed: 2013-09-26.

[44] Wikipedia. Tone mapping. https://en.wikipedia.org/wiki/Tone_mapping.

[45] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics Proceedings of ACM SIGGRAPH 2011, v. 30, no. 4, July 2011, article no. 86*, 2011.

Appendix A

Sample GP Trees

Day/Night Room:

(Top Level (Ceiling 77.4464 86.7715 18.7309 3.75074) (North Wall Center Window 79.9126 ((((95.0095) + (40.6714 + 70.6101 + 0.703375) + (41.3814 / 92.5962)) * (((93.9659 * (74.1524 * 33.8223))) - (24.3704 / 11.2108)))) (((((17.222 * 54.8905) + (59.7708) + (88.6838 - 29.045) + (57.0551 / 5.99324)) * (84.4783 - 99.5764))) + (((48.6368)))) ((((93.9986 / 70.9) - (12.7487 + 50.2278 + 19.7712)))) ((((9.07623 / (TreeValueY)) * ((TreeValueX))) - (((TreeValueX)))))) (South Wall Center Window 32.2929 ((((53.9505 + 88.4887 + 62.9813)) + (97.7846 + (16.2069) + (92.728 - 92.8943) + (76.5505 * 30.5973)) + ((10.5786 - 93.1708) * (47.1213 * 25.2851))) - (((0.461906 - 51.7645) + (25.3694 - 47.523) + (16.3485 * 74.524)))) ((((93.946 - 28.4776) - (75.5989)))) ((((56.631 / 26.7602) / (14.4037 - 0.466493))) / (((13.8856 * 91.9804) - (75.3478 * 19.9071)))) (((((TreeValueZ) - 2.74466) * ((TreeValueX))) + (((TreeValueY) - (TreeValueY)) / ((TreeValueY) - (TreeValueZ)))))) (East Wall Center Window 96.0939 ((((5.53072 * 42.3271)) / ((50.1723 / 10.6188)))) ((((24.6421 / 9.01528) * (19.7564 - 23.1581)) + ((32.8024 + 5.11876) + (47.4458 * 16.5814) + (70.4867 - 89.1084) + (79.1578))) * (((57.4096 / 90.0098)) / ((57.6777 / 32.42) / (65.2223)))) ((((74.5346 - 92.1508)) - (7.13282 * 13.4501)) + (((50.5842 * 18.842) / (76.5197 / 93.435)) + ((80.465 - 62.2127)) + ((12.9527 * 77.1304) * (58.3896)) + ((27.6001) * (85.4374 - 33.6077)))) ((((89.7264 * 28.365)) * (((TreeValueX) - 47.2481) + ((TreeValueZ) * (TreeValueZ)) + (5.45243))) - (((((TreeValueX) + (((TreeValueZ))) + ((62.9292 / 0.654863) / (TreeValueY))) / (TreeValueZ))) * ((19.3626 * 18.1387) + (((TreeValueX))) + ((TreeValueX) / 14.753) + (2.50118))))) (West Wall Center Window 79.052 ((((69.0455)) + ((51.5436) + (12.2789 / 6.34823))) / (((29.1325)))) ((((64.2993 + 31.2748 + 65.3306 + 52.9891))) - ((27.6001) + (((53.2643 + 26.4702 + 97.7846)) + (87.2546 * 32.6995) + (44.7406 + 7.72313 + 86.88)))) ((((39.6706 -

71.3153) + (24.3938 + 0.871952 + 93.2873 + 39.6053)))) (((((TreeValueY))) - (((TreeValueY) / (TreeValueX)) * (19.1995))) + (((38.8377))))) (Light Objects (Lights (((60.5906 / 98.0844))) (((9.99444 + 92.4065)) * ((8.61945 - 42.6616) + (21.5836 * 39.1376) + (62.0355 / 9.10597)))) (Lights (((55.4249) * (93.1146 - (69.0656 - 7.78403)))) 72.3592) (Lights (((50.6761 / 31.6477) - (60.8608 + 44.4131 + 19.2152)) / ((40.6367 / 47.1212) / (21.4446))) (((68.0787 + 57.3168 + 62.5107)) - ((11.5063)))) (Lights (((44.9076))) ((((98.1357) * 48.3559)))) (Lights ((((71.8001) / 87.2221) * (71.6231 * 37.1331))) (14.4037 - 0.466493)) (Lights (((62.7515) - (52.0948 + 77.767))) (((23.5826) / 72.3974) - ((38.9938 / 25.4326)))) (Lights (((27.8416 - 80.0457) / (42.7972 - 5.5161)) / (24.9532 + (59.6836 / 64.5258) + (2.24585 - (((89.3543 / 10.4078) + (62.8424 + 19.9148 + 54.9624) + (32.1461) + (20.2298 + 31.9987)) + ((1.05666 + 17.4457 + 27.5804 + 53.9584)) + ((46.3501 + 71.7097 + 68.3534) + (37.9195 / 16.8645)) + ((10.157) + (61.9147 * 12.8433) + (75.2605) + (29.219)))) + (38.2427))) (((17.2969 * 87.6796) - (38.7426 - 97.5188)))) (Lights ((50.7158 - (41.3627 / 39.8913)) / ((84.8482 - 90.9441) / (80.3299))) (((5.11811 + 10.7969 + ((17.3311 * 61.3591) - (88.7803))) * (70.5743)))) (Lights (((44.4606 / 87.4993) + (((((74.1033 * 28.5189)) * 91.9804) - (75.3478 * 19.9071)) / 98.619) + (8.96964 / 25.208) + (33.5247)) * ((79.522) / (92.1589 * 82.0261))) (((54.429) - (10.7969 - 56.8663)))) (Lights (11.5063) ((13.6454 * 93.4194) / (77.0543 / ((68.845 * 51.8702) * (6.86895 / 6.08618))))) (Lights (((29.1438) * (0.642115)) / ((74.1662 * 85.0704) - (75.8877 / 45.6951))) (((43.2209 - 18.6952) / (25.4429)))) (Lights (((97.9537 + 44.3989) + (84.8754 - 90.5615) + (48.7495 + 82.7505) + (33.1149)) * ((29.6796 * 93.4313) - (93.8133 - 74.0336))) (((43.3818 - 24.2662) / (72.3592 / 92.1586)) - ((71.3159)))) (Lights (((44.4606 / 87.4993) + (12.9421 / 98.619) + (8.96964 / 25.208) + (33.5247)) * ((79.522) / (88.5538 / 38.7062))) (((54.429) - (10.7969 - 56.8663)))) (Lights (((44.4606 / 87.4993) + ((48.7495 + 82.7505) / 98.619) + (8.96964 / 25.208) + (33.5247)) * ((79.522) / (92.1589 * 82.0261))) (((54.429) - (10.7969 - 56.8663)))) (Lights (((67.1273 / 50.8762) / (67.6874 + 86.3577 + 56.238))) (((86.6978) * (77.7398)))) (Lights (((64.9641 * 16.5663))) (((45.395) / (16.7229 * 43.4268)) - ((78.7493)))) (Lights (((17.9693) * (95.7091 / 24.7899)) / ((((11.2654 / 4.63675) - (18.8642)) / 6.13172))) (((0.881653 + 60.0205)))) (Lights (((57.9609 - 54.4517) + (72.421)) - ((63.6097 - 28.4899))) (((38.5297 - 64.9629) - (15.2346 + 68.281)) - ((73.7664 * 51.6084) + (16.8651 * 62.7648)))) (Lights (((59.3848 - 64.0994) * (96.2242 + 16.982 + 80.7223 + 60.574))) 23.9064) (Lights (((37.4221) - (18.1444 / 34.4006)) - ((53.2643 + 26.4702 + 97.7846))) (((49.6219 - 46.7475)))) (Lights (((89.0665)) + ((32.3642) / (27.0101)) + ((86.1306 - 7.40578))) (((81.535)) / ((13.6454 * 93.4194) / ((25.4429) + 23.0991 + 29.353

+ 19.9514)))) (Lights (((42.8687 + 17.0088 + 64.9419 + 88.6859)) / ((9.63035 * 66.3549) * (46.6451 - 86.9112))) (((86.5932) / (48.3104 / 29.0399)) - ((57.6674 + 30.1807)))) (Lights (((91.5592 + 37.931)) / ((77.1837 + 9.51317 + 80.1216 + 12.6586) * (19.759 * 81.4543))) (((89.3543 / 10.4078) + (62.8424 + 19.9148 + 54.9624) + (32.1461) + (20.2298 + 31.9987)) + ((1.05666 + 17.4457 + 27.5804 + 53.9584)) + ((46.3501 + 71.7097 + 68.3534) + (37.9195 / 16.8645)) + ((10.157) + (61.9147 * 12.8433) + (75.2605) + (29.219)))) (Lights (((6.54342 - 6.20009)) + ((61.7251 + 29.815 + 18.9135)) + ((11.9551 * 46.6293) + (21.8898 - 71.5604) + (69.0656 - 7.78403)) + ((49.8256 / 89.2243))) (((55.6048)) / ((53.8548 - 89.3324) - (38.471 - 51.0418)))) (Lights ((((38.7426 - (39.7235 * (46.5742))) + 53.5964)) + ((27.4212) * (95.2062))) (29.4785 / 83.2934)) (Lights (((17.3311 * 61.3591) - (88.7803))) (((67.2432 * 62.8995) - (21.5311 + 25.4703)) - ((63.9156)))) (Lights (((97.9537 + 44.3989) + (84.8754 - 90.5615) + 89.3324 + (33.1149)) * ((29.6796 * 93.4313) - (93.8133 - 74.0336))) (((43.3818 - 24.2662) / (72.3592 / 92.1586)) - ((71.3159)))) (Lights 23.9064 (((63.9727 - 29.6643) + (29.2356 / 81.1693)) + ((53.1934)) + ((91.0816 - 30.5899) + (96.673 - 72.3974) + (7.13282 * 13.4501) + (74.1662 * 85.0704)) + ((89.7797) + (71.0614 * 5.1817)))) (Lights (((8.45445 / 25.1065)) * ((11.2654 / 4.63675) - (18.8642))) (((53.2356 - 82.6649) + (82.7697)) * ((96.246 - 65.8159)))) (Lights (((86.6501 * 90.6632) + (23.9399) + (92.0712))) (((8.61297 + 33.2075 + 1.95959 + 56.238) + (45.1499 - 62.771)) * ((98.8181 / 12.2789) / ((39.4997 + 24.5617 + 79.652 + 47.8733) * (13.4413 * 10.6758))))) (Lights (((20.5385))) (((88.5538 / 38.7062) - (23.5826)))) (Lights (((31.276 + 64.4047 + 46.6955 + 34.1783) * (39.6706 - 71.3153)) / ((45.3231 + 65.772 + 20.8252) - (5.05991 + 12.4802 + 47.2602 + 20.5385))) (((71.8001))))))

Light Wall:

(Top Level (Light Wall (((((ColourValueX) * (ColourValueX)) - ((ColourValueX) + (ColourValueX) + (ColourValueX))))) (((ColourValueX))) ((((ColourValueX) + (ColourValueX) + (ColourValueX)) / (((ColourValueX) + (ColourValueX) + (ColourValueX)) * ((ColourValueX) - (ColourValueX)))) - (((ColourValueX) * (ColourValueX)) - ((ColourValueX)))) 28.3816 111.138) (Light Objects (Lights (((44.7212) / (87.5113))) (((77.015 / 19.6497)))) (Lights (((((44.7212) / (87.5113)))) / (47.1896 / (41.6061))) 98.6359) (Lights 47.1896 74.3013) (Lights ((40.9557 / 94.6789) / ((30.4136 / 21.1537) + 24.5898 + (14.9794) + 92.0992)) (((68.2847) + (1.76317)))) (Lights ((68.2847)) (((25.4459 * 61.5132)) * ((61.5132)))) (Lights (((16.5289 * 60.1051) - (58.5845)) + 71.658 + ((34.8224 / 19.5592) - (55.8135)) + ((68.6809 - 73.2503))) ((45.5883) * ((80.8613 / 66.9293) - (55.0117)))) (Lights (((29.8673 / 46.9649)) / ((29.4033 + (((24.5898 * 6.60286)) * 61.5132) + 58.0775 + 42.0276))) 46.9649) (Lights

(((84.8215))) 10.8041) (Lights 10.5352 ((43.8323 / (87.5113)))) (Lights (((87.5113))) 71.3537) (Lights ((26.1036 * (84.8215))) (((26.4665) * (37.2706)) * (((44.0774 - 95.4584))))) (Lights (((93.7039) + 72.2654 + 93.7039) + ((24.5898 * 6.60286)) + 43.8323) (98.6359 - ((58.914 * 16.0293)))) (Lights ((95.0538) - ((94.6789) - (36.6544 - 69.1817))) ((((((44.0774 - 95.4584))))))) (Lights (42.1759) 10.8041) (Lights (96.0598) (((1.434 - ((68.6809 * (40.9557 / 94.6789)) - (((58.914 * 16.0293)))))) - (85.378))) (Lights (((68.6809 * 78.3254) - (72.0553 / 71.5205)) * ((72.2654) - 43.224)) ((((58.914 * 16.0293))) * (82.9067 - (68.2847)))) (Lights 37.2995 (((85.5717 * 48.1658)) * ((((93.7039) + 72.2654 + 93.7039) + ((24.5898 * 6.60286)) + 43.8323) * (84.7575 - 88.3311)))) (Lights (56.617 - 97.1863) (((61.8143)))) (Lights (((96.6575) - 75.3718) - ((39.8562) - 43.224)) ((55.0418) / ((27.0507) - (99.4067 * (10.9423))))) (Lights ((42.1759)) 5.24751)))

Stained Glass:

(Top Level (Stained Glass (((ColourValueX) + (ColourValueX) + ((ColourValueX) * (ColourValueX))) * (((((((ColourValueX)) - ((ColourValueX))) - (ColourValueX)) + (ColourValueX) + (((ColourValueX)) - (ColourValueX))) * (((ColourValueX) + (ColourValueX) + (((ColourValueX) - (ColourValueX)) + (ColourValueX) + ((ColourValueX) + (ColourValueX) + (ColourValueX)))) / (((ColourValueX) + (ColourValueX) + ((ColourValueX) + (ColourValueX) + (ColourValueX)))))))) (((((ColourValueX) / (((ColourValueX) + (ColourValueX) + (((ColourValueX) - (ColourValueX)) + (ColourValueX) + ((ColourValueX) + (ColourValueX) + (ColourValueX)))))) / (((ColourValueX) + (ColourValueX) + (((ColourValueX) - (ColourValueX)) + (ColourValueX) + ((ColourValueX) + (ColourValueX) + (ColourValueX)))))) * ((ColourValueX) / (((ColourValueX) + (ColourValueX) + (ColourValueX)))))) (ColourValueX) 434.212 434.212))